Optimizing the Closures Development Process Using the Design Structure Matrix

by

Eric Andrew McGill

BS Mechanical Engineering, University of California at Berkeley (1999)
BA Economics, University of California at Berkeley (1999)
MS Mechanical Engineering, Stanford University (2000)

Submitted to the Sloan School of Management and the Engineering Systems Division in Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Engineering Systems
Master of Business Administration

In Conjunction with the Leaders for Manufacturing Program at the
Massachusetts Institute of Technology
June 2005

Signature of Author [illegible]
Eric A. McGill
Engineering Systems Division
Sloan School of Management
May 6, 2005

Certified by [illegible]
Daniel E. Whitney, Thesis Supervisor
Senior Lecturer, Engineering Systems Division

Certified by [illegible]
Janice Klein, Thesis Supervisor
Senior Lecturer, Sloan School of Management

Accepted by [illegible]
Richard de Neufville, Professor of Engineering Systems
Chair, Engineering Systems Division Education Committee

Accepted by [illegible]
David Capodilupo, Executive Director of Masters Program
Sloan School of Management

[This page intentionally left blank.]

Optimizing the Closures Development Process Using the Design Structure Matrix

By

Eric Andrew McGill

Submitted to the Sloan School of Management and the Engineering Systems Division in Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Engineering Systems
Master of Business Administration

Abstract

Product development processes are inherently complex sets of activities that involve a vast number of connections between participants. Engineers, designers, marketers, financial analysts, and manufacturers all have to receive information, process it, and distribute their decisions back into the system. These paths create information loops that are hidden from the participants on a long time scale and generate non-linear feedback. An analysis of the closures product and process development tasks at a major US automaker prompted the creation of new tools to optimize the ordering, identification of coupled blocks, prioritization of interactions, allocation of resources, and modeling of multiple projects. Ultimately, the analysis predicted a reduction in the average completion time of ~80%, a reduction in standard deviation of ~95%, and potential savings of ~$5B.

Unfortunately, many of the suggestions from the analysis run headlong into the organization's structural, political, and cultural environment. Structurally, the automaker is a matrix organization split along functions and program lines, constantly attempting to balance between being a strong component designer and a quality assembler. However, the functional divisions create trouble in viewing and communicating across the entire system, whether that system is the vehicle to be designed or the organization itself. Politically, the atmosphere is dominated by a strong functional orientation, authoritative traditions, and a rigid hierarchy. Culturally, the people seem to be jaded and somewhat fatalistic about the company's future. Managing change in this environment requires effort from the top and bottom of the organization, and must draw on those people inside the organization that can provide an outsider's perspective when addressing both the macro and micro challenges that will appear. Success will require using the organization against itself in order to create the initial changes that will ultimately bring about a long-term turnaround.

Thesis Supervisor: Daniel E. Whitney, Senior Lecturer
Thesis Supervisor: Janice Klein, Senior Lecturer

[This page intentionally left blank.]

Acknowledgements

I'd like to acknowledge the Leaders for Manufacturing Program for its support of my work, and the following people for their contributions...

Daniel Whitney
Jan Klein
Chip McDaniel
Tony Zambito
Pat Babcock
John Wheeler
Melissa White
Todd Dishman
Al Ver

...and the following for their incomparable contributions to my well-being...

Jenny McFadden
Chris McFadden
Maisy McFadden
Heath Holtz
Ashton Holtz

...thank you all!

Table of Contents

I. Introduction and Context

Product development processes are inherently complex sets of activities that involve a vast number of connections between participants. Engineers, designers, marketers, financial analysts, and manufacturers all have to receive information, process it, and distribute their decisions back into the system. Consequently, the work of any one participant can affect the work of many others, often through non-intuitive paths in the overall structure. These paths create information loops that are hidden from the participants, since most assume that the development process is linear. As a result, many participants and managers are surprised to find themselves caught in iterative loops where they are performing the same job because someone else "made a mistake." In reality, no mistake was made at the time. Rather, the task was finished with incomplete information and iteration was required for the design to converge. Understanding these iterative loops is absolutely fundamental to understanding the overall design process and how to improve both speed and quality of development.

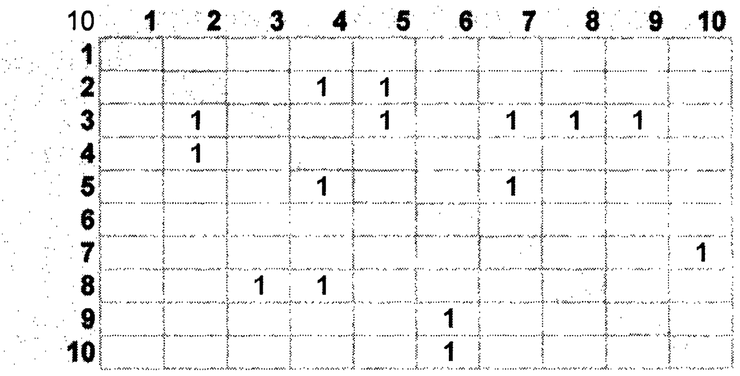
Magma Motor Company is a large US automaker that has a standing relationship with MIT to apply analytical models and tools to their processes. As part of this broader relationship, the LFM and SDM programs have performed a variety of analyses in a series of theses. Several of these past analyses have used the design structure matrix (DSM) tool to analyze the closures development process, most notably on doors and hoods. However, the number of tasks included in these matrices has typically been rather small in number, ranging from 30 to 50. In addition, the previous work has been confined to just the product development process. In this thesis, the primary intent was to combine both the product and process development, covering not only the specification of what is going to be built, but also how to build it. Once again, the area of closures was chosen to be a sufficiently complicated area for a reasonable analysis, but also as a tractable problem that would produce general results suitable for expansion to other development processes. Combining the entire closures product and process development processes generates upwards of 750 tasks, and ultimately required new methods for analyzing DSM's for computational reasons.

II. Design Structure Matrix

The design structure matrix (or DSM) is the fundamental process tool that will be used to examine Magma's closures development process.

Basic Information and Setup

The design structure matrix is a simple tool used to represent a complex design process in matrix form. Fundamentally, the DSM method assumes that a design process can be broken down into a set of individual tasks that must be completed to produce a design. However, these tasks are rarely independent from each other. Rather, the tasks are inevitably connected by a set of relationships that determine the core dynamics of the overall process. Some tasks will be precedents for future tasks, and some tasks will be dependents of previous tasks. The DSM takes these relationships and places them into a matrix form, as shown in Figure 1.



10	1	2	3	4	5	6	7	8	9	10
1										
2				1	1					
3		1			1		1	1	1	
4		1								
5				1			1			
6										
7										1
8			1	1						
9						1				
10						1				

Figure 1: Sample Design Structure Matrix

The interpretation of the DSM becomes easy with a bit of practice. Initially, the tasks are placed in symmetrical order on the rows and columns according to the current order of execution. The interior cells of the matrix are then marked with a value to describe the relationships between tasks, leaving the diagonal blank because a task cannot be dependent upon itself. In the above matrix, each of the 15 precedent and dependent relationships is marked with a 1 in the intersection cell. Precedent relationships are labeled across the rows, while dependent relationships are labeled down the columns. For example, reading across the rows reveals that task #8 requires information from tasks #3 and #4 before its work can be completed. In other words, tasks #3 and #4 are the precedents of task #8. Similarly, reading down the column reveals that task #7 delivers information to tasks #3 and #5. In other words, tasks #3 and #5 are the dependents of task #7.

In the above matrix, a quick examination of the interior reveals that there are some significant loops contained in the current order of task execution. For example, task #2 requires information from tasks #4 and #5 before its work can be completed. In a real project, this would result in task #2 being finished with incorrect or incomplete information, which would require task #2 to be redone after tasks #4 and #5 are completed later. Repeating task #2 would then require task #3 to be redone as well. Obviously, these relationships create a loop that is less than ideal, since tasks are going to be iterated with significant delays and extra tasks within the loops. It is exactly this iterative behavior that analysis of the DSM seeks to eliminate. With a little bit of thought, it's simple to realize that the ideal DSM would have all relationships *below* the diagonal of the matrix. This would result in an order of execution that delivers dependents to future tasks before they are actually needed, thereby preventing the creation of iterative loops. However, we will see that it is not always possible to push every interaction below the diagonal by simply reordering tasks. Further measures, such as deleting dependencies or adding tasks, may be required to create a purely dependent task order.

As an analytical tool, the design structure matrix is not limited to simple binary relationships in its interior. An improved DSM contains quantitative information both on the strength of interactions and on the duration of tasks, as shown in Figure 2.

10	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	5									
2		5		0.2	0.7					
3		0.1	10		0.1		0.2	0.4	0.5	
4		0.3		15						
5				0.5	20		0.3			
6						5				
7							5			0.2
8			0.8	0.7				25		
9						0.5			5	
10						0.6				10

Figure 2: Improved Design Structure Matrix

Interpretation of the improved DSM is also relatively straightforward. The numbers on the central diagonal represent the durations of the tasks in the chosen time unit. It is important to point out that the durations are for the task holding everything else constant, or assuming perfect information. Any additional time required to (re)process iterations will be taken into account during the timing calculations later. In a purely sequential process with no interaction above the diagonal and therefore no iterative loops, the total time to complete all ten tasks would be the sum of the individual times. However, if two tasks are executed in parallel, then the time would be the maximum of the two tasks. Finally, if a set of tasks must be completed in an iterative loop, the time would be the result of some unknown function of the individual times, based on the expected number of iterative loops until the tasks converge to completion. Combining all of these scenarios gives a general time to completion of

$$\Sigma(\text{sequential times}) + \text{Max}(\text{parallel times})$$

where either the sequential or parallel times could be the result of an iterative loop and a parallel time could be the sum of several sequential times. Obviously, the calculation can get quite complicated, so the details will be left for later.

The other improvement to the DSM is the inclusion of quantitative information on the strength of interactions. Giving the interior cells a value between 0 and 1 allows for a large range in describing the chance that one task will affect another. Traditionally, the value is interpreted as the percent chance that changing one task will create work for another task, either forward or backward in the process. In Figure 2, an example is that changing task #3 has an 80% chance of affecting its dependent task #8, whereas changing task #8 has only a 40% chance of affecting its dependent task #3. These values provide much more information than the previous 1's, because they reveal that the loop between tasks #3 and #8 actually has a chance of exiting at some point due to the 40% chance of task #8 returning to #3. Previously, the 100% values ensured that the project would keep cycling between tasks #3 and #8 infinitely. In fact, values less than 1 are vital to the later timing calculations, otherwise all of the iterative loops would be infinite and calculating a time would be impossible.

Previous DSM work has further augmented the level of information contained in the matrix, chiefly by breaking up the relative strength measure into smaller components. Zambito (2000) broke his task volatility into two components,

$$\text{Task Volatility} = \text{Task Sensitivity} \times \text{Information Volatility}$$

or

$$\text{Expected Value of Interaction} = \text{Probability of Change} * \text{Impact of Change}$$

where information volatility described the likelihood that information from a task would changed after its initial release, and task sensitivity described how sensitive a task would be to that changing information. Given the extreme size of the DSM that is used in the door design process, I chose to ignore this extra detail and limited myself to the single relative strength measure. However, it became necessary to estimate the split between probability and impact for the simulation of completion time.

Ordering the DSM

After an initial design structure matrix is set up, a quick glance will generally reveal that the current process order is less than optimal. More interactions are above the diagonal than are required, thereby creating unnecessary iterative loops. In addition, the tasks involved in any given iterative loop are also spaced out unnecessarily, thereby inserting extra delays in the loop and involving tasks in the loop that do not need to be there. The ordering and partitioning of a DSM seeks to resolve these issues by following 3 simple rules:

1. Place tasks with no precedents (only dependents) at the beginning
2. Place tasks with no dependents (only precedents) at the end
3. Identify coupled tasks that must be executed in a loop and group them together as closely as possible in the middle

Rules 1 and 2 are relatively straightforward to execute, since empty rows and empty columns in the DSM can be identified very easily. Once identified, it is then a simple matter to swap tasks around to place these items either at the beginning or the end of the overall process. However, the identification of the interior loops can be much more difficult. Thus far, I have found 3 separate methods for tracking the iterative loops in a DSM (http://www.dsmweb.org):

1. Path Searching
2. Powers of the Adjacency Matrix
3. Reachability Matrix

First, path searching is clearly the simplest to understand, yet one of the more annoying to handle computationally. In the path searching method, the flow of information is traced either forward or backward until a task is encountered twice. Once a set of tasks is discovered, they are collapsed into one task as long as loops do not intersect, and the process is repeated.

Second, the powers of the adjacency matrix method relies on raising the matrix to various powers to identify the loops by reading the values on the diagonal, as shown in Figure 3.

DSM

	A	B	C	D	E
A	0	0	1	1	0
B	0	0	0	0	1
C	1	1	0	0	1
D	0	1	0	0	0
E	0	0	0	1	0

Square

	A	B	C	D	E
A	1	1	0	0	1
B	0	0	0	1	0
C	0	0	1	1	1
D	0	0	0	0	1
E	0	1	0	0	0

Cube

	A	B	C	D	E
A	0	0	1	1	1
B	0	1	0	0	0
C	1	1	0	1	1
D	0	0	0	1	0
E	0	0	0	0	1

Figure 3: Example of Adjacency Matrix Method

As is shown by the highlighted cells, the squared matrix reveals that tasks A and C are in a two step loop. Similarly, tasks B, D, and E are in a three-step loop. Additional powers can be calculated to find higher order relationships, but none are revealed in this example. Once all of the step loops are identified, the DSM can then be ordered and partitioned.

Third, the reachability method treats the DSM as a multi-level hierarchy that ranks tasks by how many "levels" down in the DSM they are. The top level tasks are independent from the other tasks by the virtue of having no precedents. After top level tasks are identified, they are removed and the next top level of the sub-matrix is identified. The ultimate order is then reassembled by using successive top levels. The process takes the following form:

1. Construct a table with four columns and give each diagonal element of the DSM a value of 1.
    a. In the first column, list all the elements in the matrix.
    b. In the second column, list the set of all the input elements for each row in your table.
    c. In the third column, list the set of all output elements for each row in your table.
    d. In the fourth column, list the intersection of the input and output sets for each element in your table.
2. Identify top level elements and remove them and their connections from the table. An element is in the top level hierarchy of the matrix if its input (row) set is equal to the intersection set.
3. Go to step 1.

Several of these tools are available on the web either in Java form or as downloadable spreadsheets. In initial testing, all of them produced the same order and appeared to have no trouble executing any DSM given to them. However, all of the tools took an incredibly long time to order the DSM, ranging from a few seconds for a 10 by 10 matrix to few minutes for a 50 by 50 matrix. Considering that the first DSM for the door design process was 773-by-773, the initial estimate of calculation time (increases as n^2) was ~15 hours. This time was unacceptable, so I had to come up with an improved algorithm to greatly reduce the computational time required to generate a reordered DSM. Furthermore, the most capable tool was limited to a 255-by-255 matrix because it was written in Microsoft Excel.

The New Algorithm

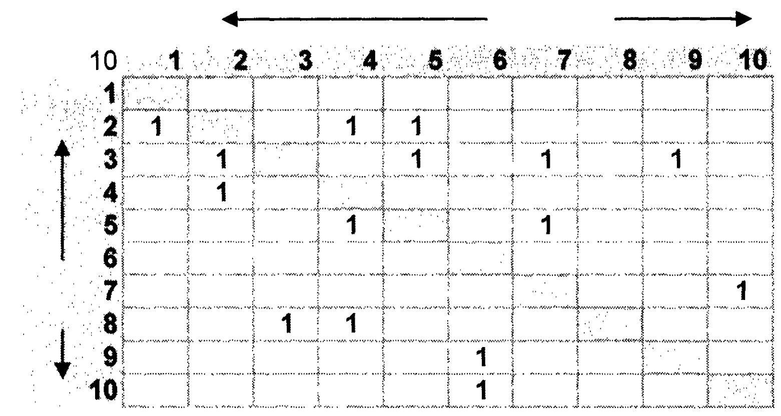
When I began breaking down the code in the Excel-based tool, it quickly became apparent that the code was not optimized for computational speed. The spreadsheet was using just one of the DSM methods (the reachability matrix) and applying it to the entire matrix without searching for empty rows and columns first. This required Excel to raise an n-by-n matrix to the n^{th} power by hand. Even with the best computational tricks, Excel is simply not well equipped to deal with this level of matrix mathematics. After deconstructing the existing code, I came up with three significant computational improvements:

1. Use a much more capable platform, such as Matlab, for the matrix manipulation.

2. Follow the complete 3-step DSM optimization process and move empty rows and columns before identifying couplings.
3. Iterate the 3-step process after finding *top level tasks only* in the coupling step.

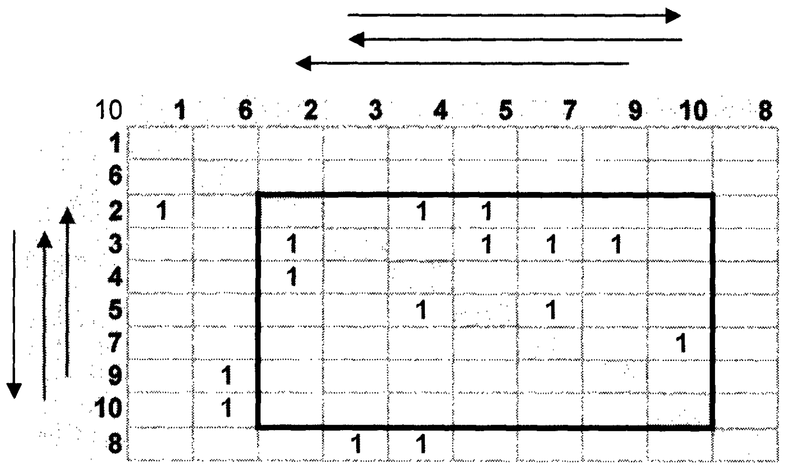
The first recommendation is relatively straightforward, since Matlab can handle matrices well above 255-by-255 and can process a 773-by-773 matrix to the 773th power in a few seconds versus several hours for Excel. The second recommendation appears to correct an oversight in the creation of the original code, since searching for empty rows and columns as a first step is an easy way to reduce the size of the matrix that needs to be raised to a power. However, it's quite possible that the original authors never anticipated dealing with matrices larger than 100-by-100, so the computational difficulties may not have been as important. Regardless, not searching for empty rows and columns in a large sparse matrix unnecessarily increases the computational time by several orders of magnitude. The third recommendation came from an insight into the reachability matrix method. After the first set of top level tasks are identified and removed from the matrix, the original method simply proceeds to do the same set of calculations on a smaller matrix and identify the second level of tasks. Again, this fails to examine the smaller matrix for empty rows and columns before proceeding on to the coupling identification. From a little bit of experimentation, it was clear that removing the top level tasks and their connections would generally create a set of empty rows and columns in the remaining matrix. This would then further simplify the tasks and result in an even smaller matrix for the next reachability calculation. With these three recommendations, I created an iterative process that was able to quickly identify and group sets of independent and coupled tasks.

In general, the new algorithm is a purer implementation of the original 3-step DSM optimization process, simply designed to search for empty rows and columns more often while using the reachability matrix method to identify coupled tasks. Since empty rows are written to the front of the order, empty columns to the back, and the top level tasks to the middle, the main challenge of the new algorithm was keeping track of new positions and extracting the correct portion as the next sub-matrix for further processing. However, by writing from both ends toward the middle, the new algorithm was able to further improve the processing speed. Graphically, the process looks as follows:



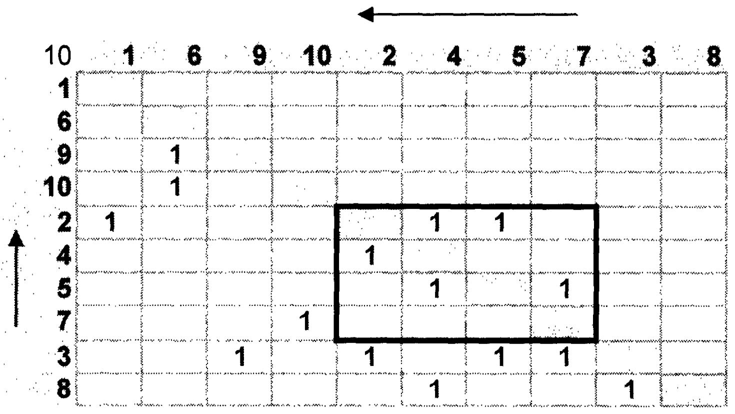
10	1	2	3	4	5	6	7	8	9	10
1										
2	1			1	1					
3		1			1		1		1	
4		1								
5				1			1			
6										
7										1
8			1	1						
9						1				
10						1				

Figure 4: Identify and move empty rows 1 and 6, and empty column 8



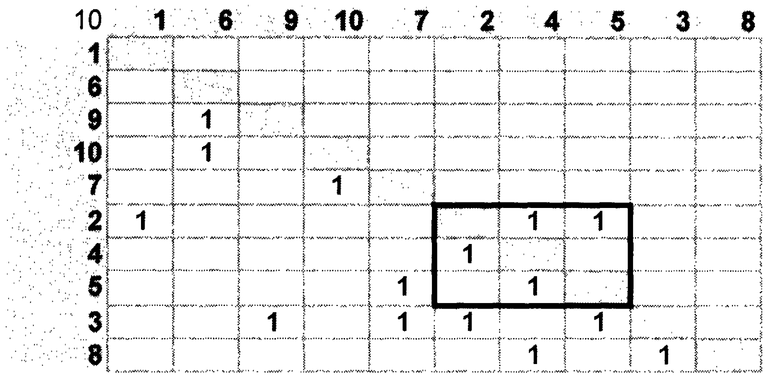
10	1	6	2	3	4	5	7	9	10	8
1										
6										
2	1				1	1				
3			1			1	1	1		
4			1							
5					1		1			
7									1	
9		1								
10		1								
8				1	1					

Figure 5: Mark sub-matrix, then identify and move empty rows 9 and 10, and empty column 3



10	1	6	9	10	2	4	5	7	3	8
1										
6										
9		1								
10		1								
2	1					1	1			
4					1					
5						1		1		
7				1						
3			1		1		1	1		
8						1			1	

Figure 6: Mark sub-matrix, then identify and move empty row 7



10	1	6	9	10	7	2	4	5	3	8
1										
6										
9		1								
10		1								
7				1						
2	1						1	1		
4						1				
5					1		1			
3			1		1	1		1		
8							1		1	

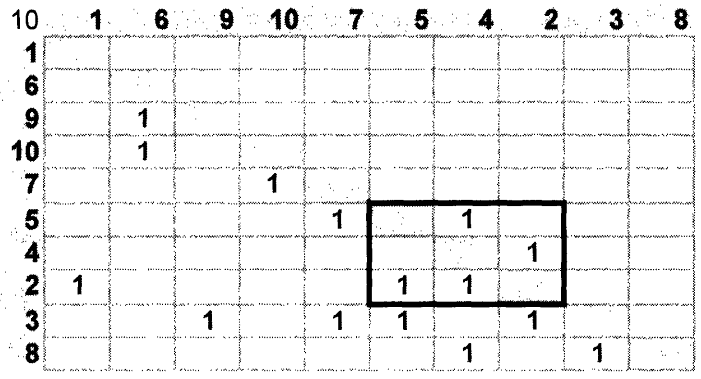
Figure 7: Mark sub-matrix, then identify tasks 2, 4, and 5 as coupled top level tasks using reachability

At this point, the original DSM optimization tools stopped. In fact, this is the exact order that is output from the Excel based tool for this particular example. However, much more can be done with the ordering information. Up until this point in the analysis, there has been nothing to distinguish one DSM from another in terms of quality. Changing the order of the tasks does nothing to change the core relationships, so eigenvalues cannot be used to separate one DSM from another. Also, calculating a completely parallel execution time does not produce a distinction for ordered DSM's, because all of the ordered DSM's have grouped the tasks in the same way and all tasks are assumed to be executed at the same time. The only meaningful difference between ordered DSM's can be seen in a calculation of the sequential times, assuming that tasks are done in order and repeated as necessary. An even better comparison would be generated by running an actual simulation of the task execution.

Assuming sequential execution, my hypothesis was that minimizing the weighted sum of "distances" from the diagonal to each feedback interaction is a meaningful optimization criterion. This minimization criterion also has some significant organizational interpretations. Since it is rare for an organization to run tasks purely in parallel (e.g. a group working together in a big conference room), especially if the tasks are fairly general and the teams are large, the sequential case must be considered for large-scale project management. In this scenario, minimizing the sum of weighted distance above the diagonal means that each instant of rework will cover a shorter distance "back in time" to the previous tasks. This means that fewer tasks will need to be repeated and the time to completion will be reduced. Mathematically, the following optimization equation is being used,

$$Minimize \sum_{i} \sum_{j>i} w_{ij} \cdot d_{ij}$$

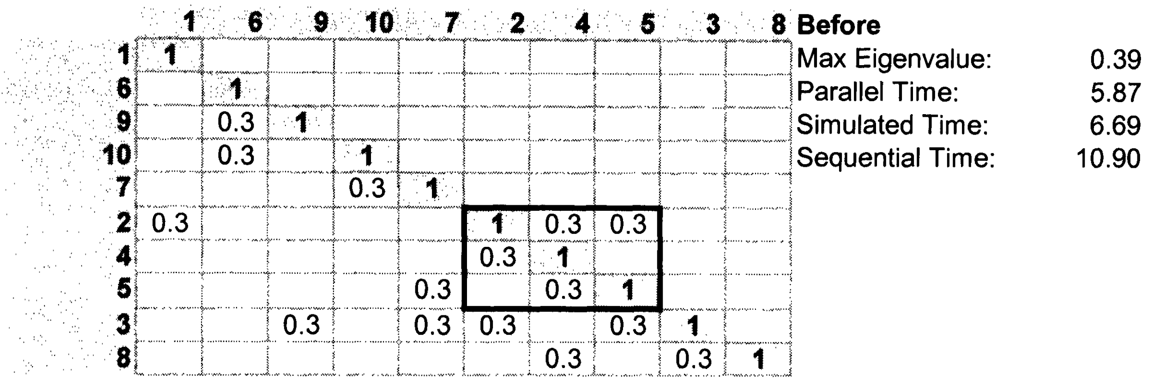
where w_{ij} is the interaction strength and d_{ij} is the distance from the diagonal. The minimization is done by swapping tasks in the sub-matrix until the smallest aggregate distance is found. Because of the weighting, the minimization will tend to bring the stronger interactions toward the diagonal, where their more frequent feedbacks will cover fewer tasks. After running the swapping routine on the example DSM, the output is the following:



10	1	6	9	10	7	5	4	2	3	8
1										
6										
9		1								
10		1								
7				1						
5					1		1			
4								1		
2	1					1	1			
3			1		1	1		1		
8							1		1	

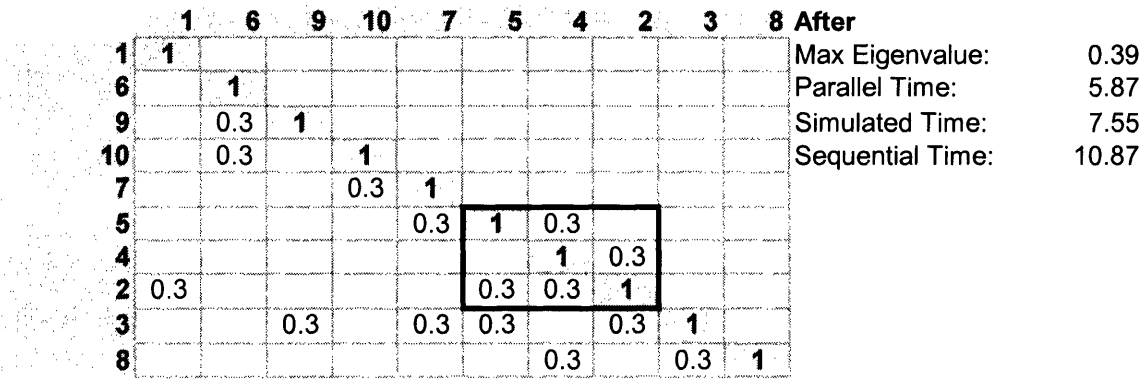
Figure 8: Optimized DSM through swapping

As can be seen, the previous order of 2 4 5 is now switched to 5 4 2. This reordering does absolutely nothing to affect the core relationships (or eigenvalues) of the matrix, but it does create two 1-step feedbacks instead of one 1-step and one 2-step in the previous solution. This further reordering changes the process so that completing the third task in the sub-sequence does not automatically require reworking the first. In other words, two 1-step feedbacks give the process an extra chance to escape rework. Of course, the above example has 100% chance of feedback, so completion is not possible in any case. To better illustrate, an example of the differences using all 30% chances follows:



	1	6	9	10	7	2	4	5	3	8
1	1									
6		1								
9		0.3	1							
10		0.3		1						
7				0.3	1					
2	0.3					1	0.3	0.3		
4						0.3	1			
5					0.3		0.3	1		
3			0.3		0.3	0.3		0.3	1	
8							0.3		0.3	1

Before	
Max Eigenvalue:	0.39
Parallel Time:	5.87
Simulated Time:	6.69
Sequential Time:	10.90



	1	6	9	10	7	5	4	2	3	8
1	1									
6		1								
9		0.3	1							
10		0.3		1						
7				0.3	1					
5					0.3	1	0.3			
4							1	0.3		
2	0.3					0.3	0.3	1		
3			0.3		0.3	0.3		0.3	1	
8							0.3		0.3	1

After	
Max Eigenvalue:	0.39
Parallel Time:	5.87
Simulated Time:	7.55
Sequential Time:	10.87

Figure 9: Comparison of DSM results after minimization of distances

For the above timing calculations, a quick summary of the terms is useful:

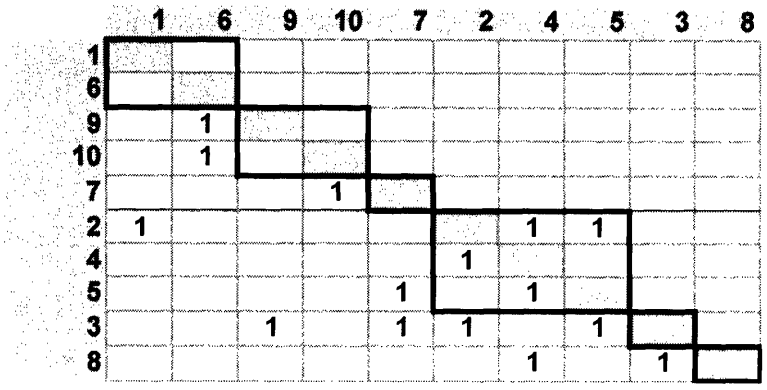
1. *Parallel Time:* Assumes all tasks are worked in each time period until work is complete, and can be used as a lower bound.
2. *Sequential Time:* Assumes tasks are worked in their stated order, and follows a Markov chain of paths until the end is reached. Typically, this calculation is performed using signal flow graphs and can be used as an upper bound.
3. *Simulated Time:* Performs an actual simulation of the work, beginning each task when its upstream precedents are complete and generating rework based on interaction probabilities. Times are distributed between the upper and lower bounds, and can be viewed as the best representation of the actual completion times.

As shown in the above comparison, there is a slight improvement in the sequential time when the algorithm minimizes the weighted distance above the diagonal. In this particular case, the improvement is negligible, but comes for free since the maximum eigenvalue and parallel time calculations do not change. Unfortunately, the simulated completion time shows a remarkable increase in the second matrix, largely due to the extra wait that task 2 experiences. In the first matrix, task 2 can begin after task 1 completes since there is only one upstream precedent. However, the second matrix gives task 2 three upstream precedents (tasks 1, 5, and 4) and therefore requires task 2 to wait until all three are complete to begin. This delay propagates through the simulation and ultimately is responsible for the increased simulation time. Further testing with other sample matrices revealed a similar trend for the difference between the swapped and un-swapped matrices, thereby causing me to reject my hypothesis about minimizing the weighted sum of distances above the diagonal. Consequently, the new algorithm will not be using the swapping operation.

The final step in the new algorithm is establishing the relevant blocks of activities and reducing the matrix into a viewable form. In the process of running the algorithm, the code builds the final DSM order as follows:

$$[f_{1(1-?)}\ m_1\ f_{2(1-?)}\ m_2\ f_{3(1-?)}\ m_3\ \ldots.\ b_{3(1-?)}\ b_{2(1-?)}\ b_{1(1-?)}]$$

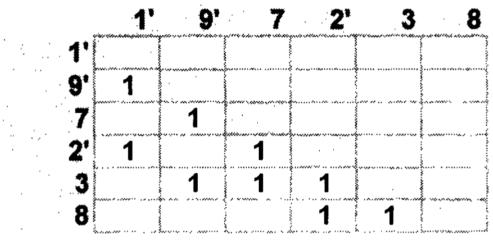
where each letter represents a series of tasks that was pulled to the front, back, or middle of the new order. The second subscript on each front and back series indicates that each major block can have several smaller series pulled to the front before work starts on the middle series. Each of these smaller series is a set of tasks that can be executed in parallel. In the case of the front and back series, the tasks are all independent of each other – no precedents or dependents within the smaller series – thereby allowing the tasks to be executed in parallel and to be completed in the maximum individual time. However, f_{11} will almost surely have dependents in f_{12} or m_1, so f_{11} must be completed before those following blocks can start. In the case of the middle series, the tasks are a coupled set of tasks that must be executed in parallel and complete a number of iterations before convergence. Therefore, the completion time for middle block follows a simulation or parallel function of the individual task times and interaction. After calculating all of the parallel times, the total time for the process is the simple sum of the parallel blocks executed sequentially. Graphically, the blocking looks as follows:



	1	6	9	10	7	2	4	5	3	8
1										
6										
9		1								
10		1								
7				1						
2	1						1	1		
4						1				
5					1		1			
3			1		1	1		1		
8							1		1	

Figure 10: Blocked DSM to show tasks executed in parallel

In concept, these blocks are quite similar to the banding discussed in *Browning, 2002*. These blocks are excellent gates for the organization in the real process, because each block should be completed before work begins on the next block. In fact, it makes sense to have organization structured around these blocks, with sign-offs and approvals located at the end of each major iterative interaction. These sign-offs should be quite hard, so as to not allow a non-converged design to pass through. If a non-converged design passes the gate, then its subsequent completion will force changes in other blocks and essentially destroy the entire benefit of ordering and blocking in the first place. For this reason, a simple process chart or check-off sheet is simply not enough to enforce compliance with the iterative blocks. Rather, system integrators should be assigned to these blocks with the responsibility and authority to manage the convergence. In reality, system integrators might also need to be able to restrict access to data before and after the current block so that the structure isn't violated. Ultimately, each of these blocks can then be reduced to an equivalent single task representation to simplify the interpretation of the DSM:



	1'	9'	7	2'	3	8
1'						
9'	1					
7		1				
2'	1		1			
3		1	1	1		
8				1	1	

Figure 11: Reduced DSM

In this representation, each group of tasks has been reduced to a single task by simulating the duration and combining the interactions. Interactions were taken to be the average of the non-zero items across the rows and columns. From examining the reduced matrix above, it's clear that the reduced DSM is a purely sequential process with individual tasks representing the condensed parallel execution of a series of tasks. Of course, the diagonal values would contain the new times instead of the original independent task times.

Computationally, the new algorithm has vastly outperformed the original DSM tools. In a straight calculation of an order, the Excel-based code processed the base 773-by-773 matrix in 3 hours on a standard 2.4 Ghz PC with some input and output tricks across worksheets. On the same machine, the new algorithm took 2-3 minutes to produce the same final order with the swapping and minimization of distance turned off.

Sensitivity and Randomization

Much of the data inherent to a DSM is difficult to accurately measure and is frequently accompanied by a high standard deviation. For these reasons, initial data such as durations and interaction strengths must be taken as averages of triangular distributions. For computational purposes, I created a separate SD matrix to track and combine as the core DSM was reordered. The basic rules for creating the standard deviations were:

$$\begin{aligned} SD_{duration} &= 0.25 * duration \\ SD_{strength} &= 0.15 * strength \end{aligned}$$

Equation 1: Standard deviation conventions

As a rough proxy, these SD's gave an excellent representation of the sensitivity of the DSM to changes in the durations and interaction strengths. For the example DSM, the starting SD matrix follows:

10	1	2	3	4	5	6	7	8	9	10
1	0.25									
2	0.05	0.25		0.05	0.05					
3		0.05	0.25		0.05		0.05		0.05	
4		0.05		0.25						
5				0.05	0.25		0.05			
6						0.25				
7							0.25			0.05
8			0.05	0.05				0.25		
9						0.05			0.25	
10						0.05				0.25

Figure 12: Starting SD matrix for example DSM

Timing Calculations

Although the parallel and sequential methods of calculating time to completion have been mentioned up to this point, further detail is useful in understanding exactly what assumptions and methods are used.

Sequential Time *(Smith and Eppinger, August 1997)*

In the sequential model of completion, each task is assumed to be completed in order with a finite probability of affecting the other tasks. In the example DSM below, task C has a 50% chance of affecting task B, while A has a 40% chance of affecting task B.

	A	B	C
A	4	0.2	
B	0.4	7	0.5
C	0.3	0.1	6

Figure 13: 3-Stage example

This method also assumes that there are no errors or delays in the execution of any task. If these were present, the DSM would have to be adjusted appropriately. In reality, this DSM is a matrix representation of a Markov chain that leads to completion of the process after task C is finished. This Markov chain can be completely drawn as follows:

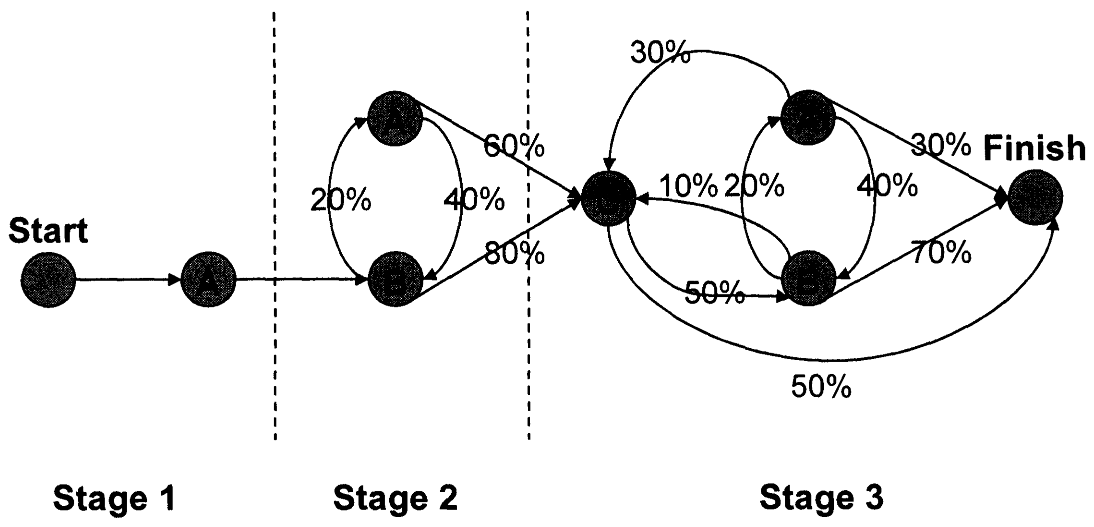


Figure 14: Complete Markov chain for 3-stage example

The time is calculated by proceeding backwards through the chain – first finding the expected time in node C at stage 3, then B at stage 2, then A at stage 1. For node C in stage 3, a set of linear equations can be written to describe the expected time:

$$\begin{aligned} r_A &= 0.4r_B + 0.3r_C + 4 \\ r_B &= 0.2r_A + 0.1r_C + 7 \\ r_C &= 0.5r_B + 6 \end{aligned}$$

Equation 2: Markov linear equations

where the equations come from the columns of the DSM and charge each node with its base task time and the weighted average of its dependents. Solving this set of equations reveals that the time r_C is 11.21. Similar sets of equations can be constructed for node B and A in stages 2 and 1, respectively. Solving these sets reveals r_B to be 8.48 and r_A to be 4. Summing the individual times gives a total time of 23.69. This method can also be reduced to a simple matrix computation that forgoes the construction of many sets of linear equations (*Eppinger and Smith, August 1997*).

Signal Flow Graphs *(Eppinger, Nukala, and Whitney, 1997)*

Signal flow graphs are well known tools for circuit analysis and discrete event systems. The graphs begin with a similar depiction of the Markov chain shown in Figure 14, except the probabilities are replaced with z-transform functions.

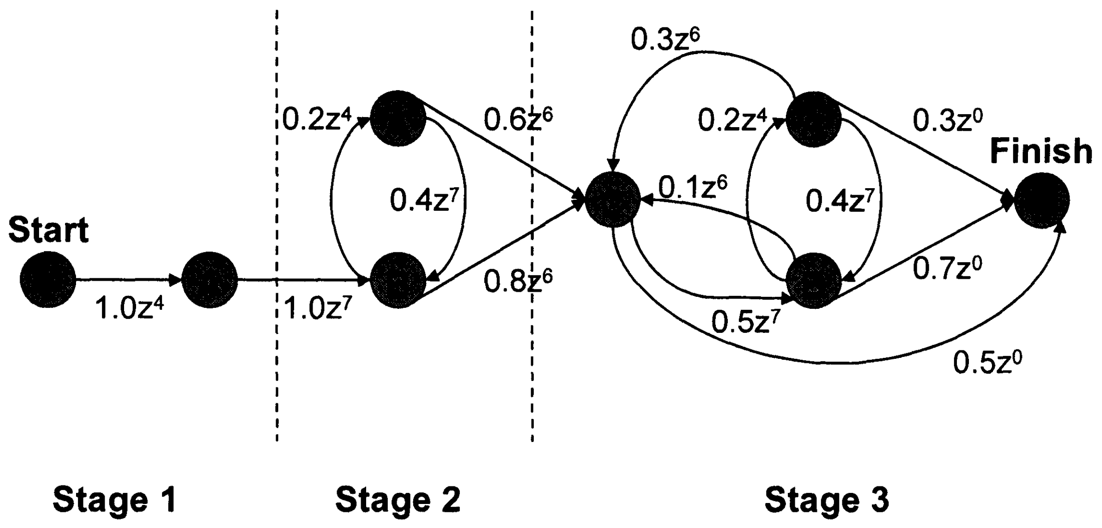


Figure 15: Example Markov chain with z-transforms

For each branch of the Markov chain, the z-transform coefficient represents the probability of traveling down that branch and the z-transform exponent represents the time used to traverse the branch. In general, the z-transform simplifies the math required in calculating the overall transmission time, since the coefficients will be used to multiply the probabilities and the exponents will be used to add the times.

The simplification of the signal flow graph follows four simple rules to develop the system of equations:

1. Signals travel only in the direction of arrows
2. A signal traveling along a branch is multiplied by the transmission value of that branch
3. The value of any node is the sum of all signals entering that node
4. The value of any node variable is transmitted along all branches leaving that node

Once the system of equations is written down, the equations can be combined to successively eliminate variables until a generic transform function between the start and finish nodes is found. For the example shown above, the transform equation between the start and finish is:

$$T_{sf} = -z^{17} \frac{3z^{15} - 85z^{11} - 150z^{4} - 700z^{7} - 1000}{2500 - 400z^{11} - 125z^{13} - 75z^{17} + 10z^{24} + 6z^{28} + 16z^{22}}$$

Equation 3: Transform for 3-stage example

This transform equation contains all of the information to generate a histogram of transmission times, but in practice only the mean and variance will be used. It is relatively simple to calculate the mean and variance using the following equations:

$$E(L) = \left.\frac{dT_{sf}}{dz}\right|_{z=1}$$

$$Var(L) = \left.\frac{d\left(z\frac{dT_{sf}}{dz}\right)}{dz}\right|_{z=1} - E(L)^2$$

Equation 4: Expected value and variance from transform equation

Following these equations, the mean of the transmission time is 23.69, and the variance is 62.27, implying a standard deviation of 7.89. As you will note, the mean time calculated here is exactly equal to the standard sequential time calculated above. Since the new DSM algorithm assumes that all iterative blocks are executed in parallel for organizational purposes, the sequential calculation is only used at the end of the timing calculation when everything is a series of parallel blocks. For this reason, I chose to use the sequential time calculation presented from *Smith and Eppinger, August 1997* for computational simplicity.

Parallel Time *(Smith and Eppinger, March 1997)*

In the parallel model of completion, all tasks are assumed to be worked on at the exact same time. However, the DSM is treated as a work transformation matrix that will tend to create some work (or rework) for another task in the process. This implies that the process will eventually converge to a solution over time as the percentages create a steady state. This behavior can be represented as a changing work vector u.

$$\begin{aligned} &u_{t+1} = Au_t \\ &or \\ &u_t = A^t u_0 \end{aligned}$$

Equation 5: Work transformation matrix relationship

where A is the matrix that describes the pattern of rework and u_0 is a vector with the original task times. A is essentially the DSM matrix with the diagonal elements set to zero, and u is the diagonal of the DSM in vector form. The total work vector U is then the sum of all the individual u_t's from each time to completion. If A has linearly independent eigenvectors, then it can be decomposed, and the solution can be found as the time intervals approach infinity.

$$\begin{aligned} &A = S\Lambda S^{-1} \\ &A^t = S\Lambda^t S^{-1} \\ &U = \sum_t A^t u_0 = S\left(\sum_t \Lambda^t\right)S^{-1}u_0 \\ &\lim U = S(I-\Lambda)^{-1}S^{-1}u_0 \end{aligned}$$

Equation 6: Completion time calculation

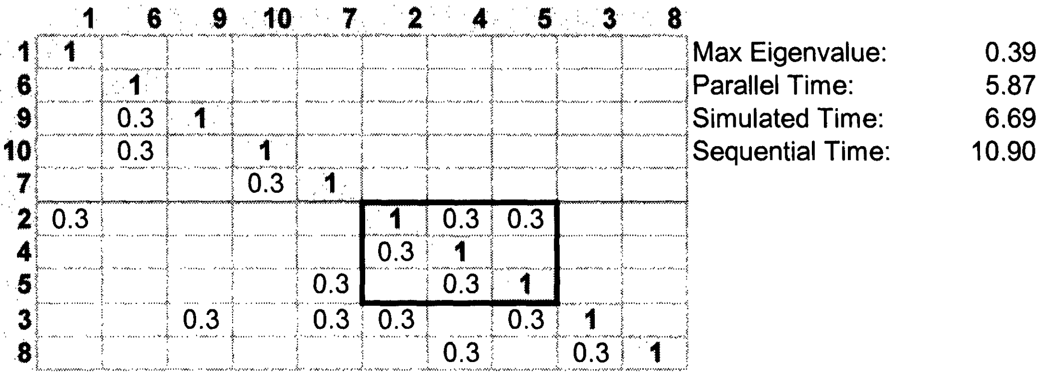
The completion time for the process is the maximum of the work vector U since the operation is occurring in parallel. Calculation of the eigenvalues and eigenvectors also allows for the determination of the dominant modes and convergence limits *(Smith and Eppinger, March 1997).*

Simulation *(Browning, 2002)*

Unfortunately, each of the above timing calculations has a variety of shortcomings. The parallel time calculation assumes that every task can be worked in each time period, which is unrealistic both from an information and resource perspective. The sequential and signal flow calculations are fundamentally using an OR-gate assumption by allowing a task to begin if any of its precedents sends a signal. In reality, all of the upstream precedents are required to be complete before a task can begin. As a consequence of these inherent assumptions, the parallel and signal flow times become lower and upper bounds on the true completion time distribution. Thus, simulation is the only way to properly identify the distribution, with each run calculated as follows *(Browning, 2002)*:

1. Generate task duration times from a triangular distribution of the minimum, most likely, and maximum times for each task as the initial work values
2. Work tasks that have work remaining and all of their upstream precedents complete
3. For each completed task, generate rework based on the rework probability, impact, and learning curve
4. Loop steps 2 and 3 until there is no work remaining

By running the simulation a number of times, a distribution of completion times can be obtained. In practice, these times are a blend of parallel and sequential task execution and fall between the previously described upper and lower bounds. In addition, the distributions tend to be long-tailed to the right, suggesting that there are finite chances for quite long completion times. For the example matrix, the following gives a summary of the timing calculations with the learning curve turned off:



	1	6	9	10	7	2	4	5	3	8
1	1									
6		1								
9		0.3	1							
10		0.3		1						
7				0.3	1					
2	0.3					1	0.3	0.3		
4						0.3	1			
5					0.3		0.3	1		
3			0.3		0.3	0.3		0.3	1	
8							0.3		0.3	1

Max Eigenvalue:	0.39
Parallel Time:	5.87
Simulated Time:	6.69
Sequential Time:	10.90

Figure 16: Timing Calculations for the Example DSM

In addition, the completion time distribution for 500 simulations is as follows, using a probability of 30%, an impact of 100%, and no learning:

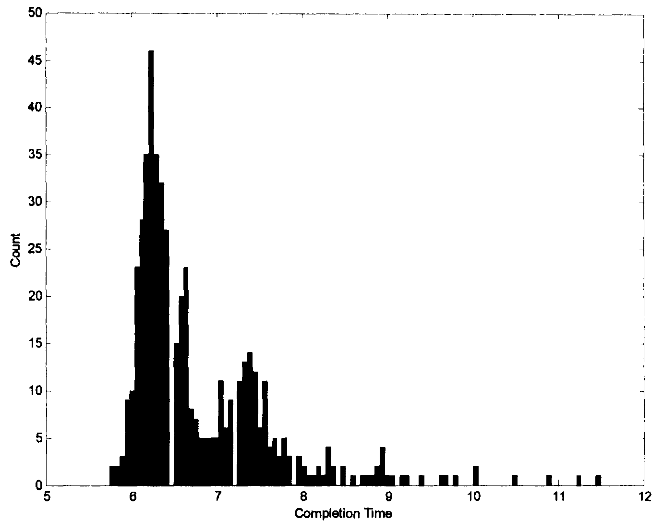


Figure 17: Simulation Results for the Example DSM

As the image shows, the distribution is a right-hand long tail because of the continuous possibility for generated rework. Overall, the distribution appears to be lognormal with a fixed minimum (i.e., a lognormal distribution offset by the minimum of the actual distribution). Modeling the distribution in this fashion will prove to be quite useful in the future calculations. Activating the learning curve such that task times are decreased by 40% if executed more than once gives the following results:

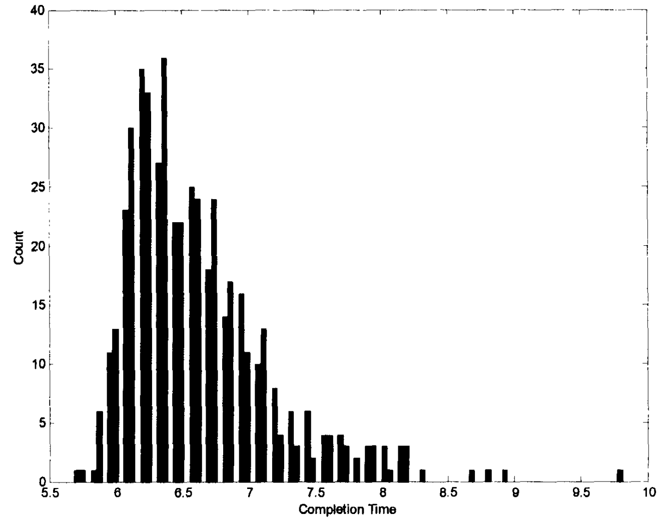


Figure 18: Simulated DSM with learning of 40%

As can be seen, the addition of learning primarily truncates the right-hand tail of the distribution because extremely long outcomes are less likely. For the rest of the calculations in this thesis, the simulation model will be used.

Simulation Extension

Although the basic simulation method is quite useful for estimating completion times for a single project, it does not address resource constraints, utilization rates, or multiple projects. Thus, the model was extended to cover all of these cases.

Resource Constraints

For resource constraints, a simple matrix was created to outline the resource requirements for each task:

Task	A	B	C
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
10	1	1	1

Figure 19: Sample Resource Requirements for the Example DSM

In the above table, each entry represents the amount of each resource type (A, B, or C) required to begin work on the given task. These resource requirements are then matched against the level of available resources to determine whether or not work can begin. Obviously, the number of resource types can be extended as necessary. Other than this tracking of resources in use, the extended model is not materially different from the Browning model. To illustrate the effect of resource constraint, the example matrix was simulated with the minimum amount of resources available to actually run (1 for each of resource type A, B, and C):

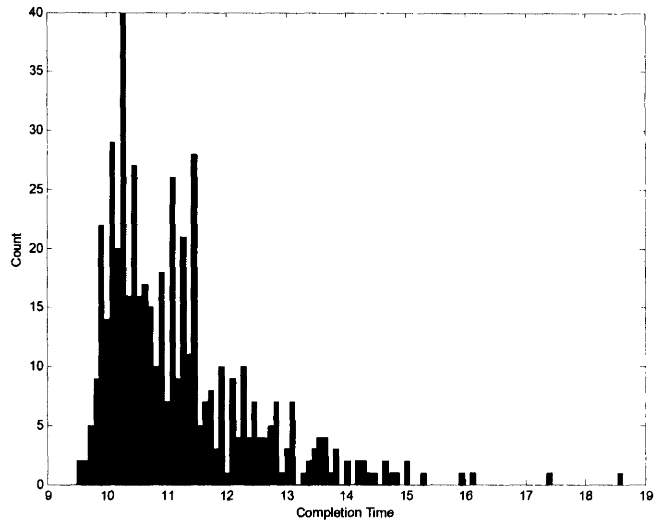


Figure 20: Simulation Results for Resource-Constrained Example DSM

As the distribution illustrates, the resource constraints have a dramatic effect on the completion time – moving the most likely time from ~6.5 in Figure 17 to ~10.5 days in Figure 20. Essentially, there is no parallel activity occurring at all because only the minimum amount of resources is available.

In the course of simulating the DSM under resource constraints, it is relatively easy to track the resource needs and subsequently relax the model until there are no resource needs during the simulation. Ultimately, a trade-off is created between having more resources available for the duration of the project and reducing the time to complete the project. Graphically, this ends up being a cost curve with resources available * time on the y-axis and completion time on the x-axis. The following graph shows the simulation results of relaxing the example DSM from a starting point of 1 for each resource type:

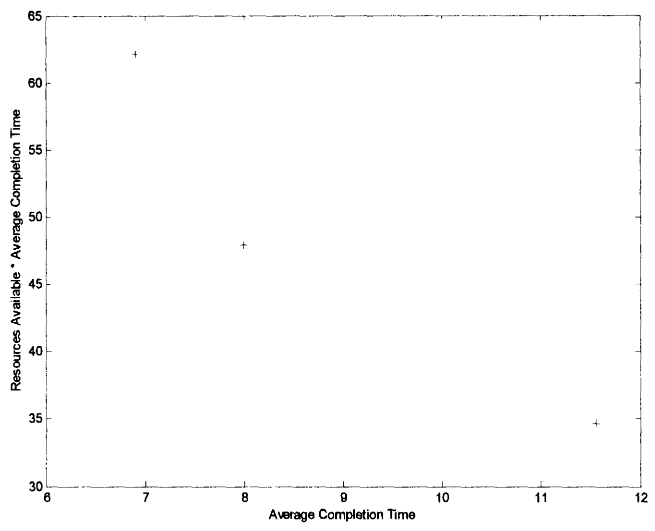


Figure 21: Resource Relaxation for the Example DSM

As the above figure shows, applying more resources to the DSM will reduce the completion time, but at the expense of paying much more for those resources to be available over the duration. Naturally, this trade-off creates a curve that can be optimized through the use of a budget constraint to find the appropriate tangent point:

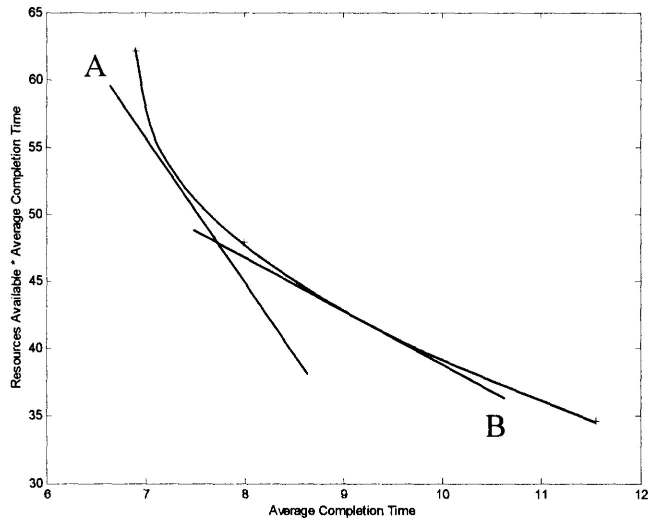


Figure 22: Budget Constraints Applied to Resource Curves

As the above figure shows, a budget constraint can be used to estimate the optimum level of resources to have available for a given project. If a company values time more and uses budget constraint A, then they will choose to allocate a higher level of resources, incur a higher cost, and receive a completion time of ~7.3 days in return. However, if a company values time less and uses budget constraint B, they will choose a lower level of resources, a lower cost, and receive a completion time of ~8.2 days. Each choice is perfectly valid, but should be made with a specific value of time in mind.

Utilization Factors

Next, the model was extended to include a concept of utilization by introducing a factor that adjusted the amount of resources applied to the tasks. For example, a factor of 1.5 would multiply the resource requirement by 1.5 but also divide the task duration by 1.5. Naturally, this assumes that tasks are completed by finishing a given amount of person-days of work and that there is a trade-off between days and people. However, the reality is that a finite amount of time is required to complete a task in order to prevent a situation with infinite resources and zero completion time. Thus, the factor is limited to a maximum of two, or a minimum duration of 50% of the original specification. In the simulation, the intent is to determine whether splitting resources between tasks and working each slower or dedicating them to work one task faster is more effective in reducing the completion time. The following chart shows the progression of the simulation from a factor of 0.5 to 1, using a base resource level of 1 for each resource type:

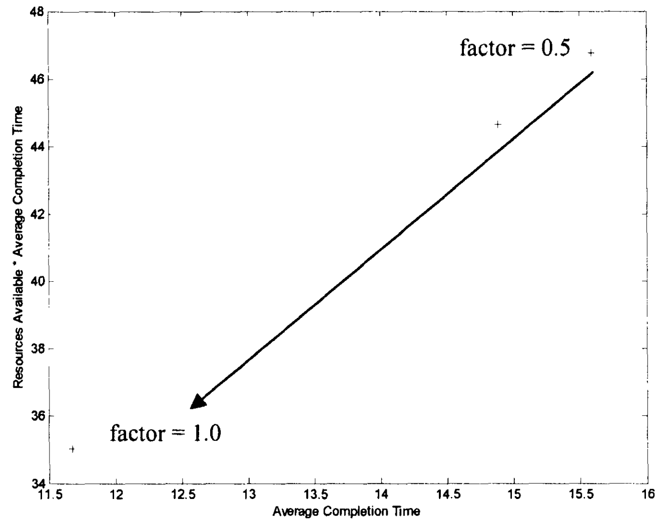


Figure 23: Simulation Results for a Variety of Factors

In general, the above figure shows a general downward progression as the factor increases from 0.5 to 1.0. This suggests that dedicating resources and speeding the execution of individual tasks is the preferred course of action, at least until the resource constraint is reached at a factor of 1.0. If the resource relaxation routine is then run after reaching the resource constraint at the farthest downward and leftward point, the complete picture is as follows:

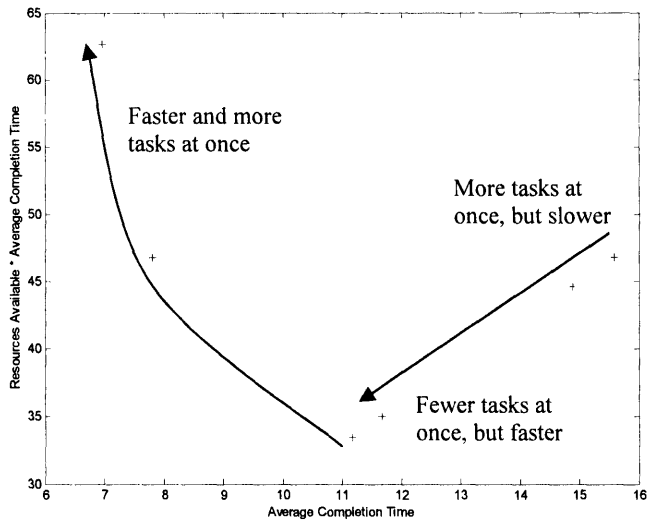


Figure 24: Factor Progression and Resource Relaxing for Example DSM

As the above figure shows, relaxing the resource constraint after reaching a factor of 1 creates a similar trade-off as illustrated in Figure 21. Again, this develops a frontier that can be used with a budget constraint to optimize the resource allocation. However, combining the factor behavior and resource relaxation creates an interesting phenomenon. If the prescribed action is to always use the highest factor possible until the resource limit is reached and then to relax the resource limit, why wouldn't this behavior continue forever? I.e., once relaxing the resource limit, the factor should increase again until the new resource constraint is reached, then the resources limit is relaxed once more. In this pattern, the simulation should continue looping down the cost and completion time indefinitely. However, the simulation here contains an internal limit of task durations – namely 50% of the original duration. Consequently, a final frontier will be reached when the tasks are being completed as fast as they can, which happens to be 50% of their initial duration in this simulation. Any additional resources added after this point will merely increase the cost. Thus, running the simulation over a variety of resource levels and factors produces the following master chart:

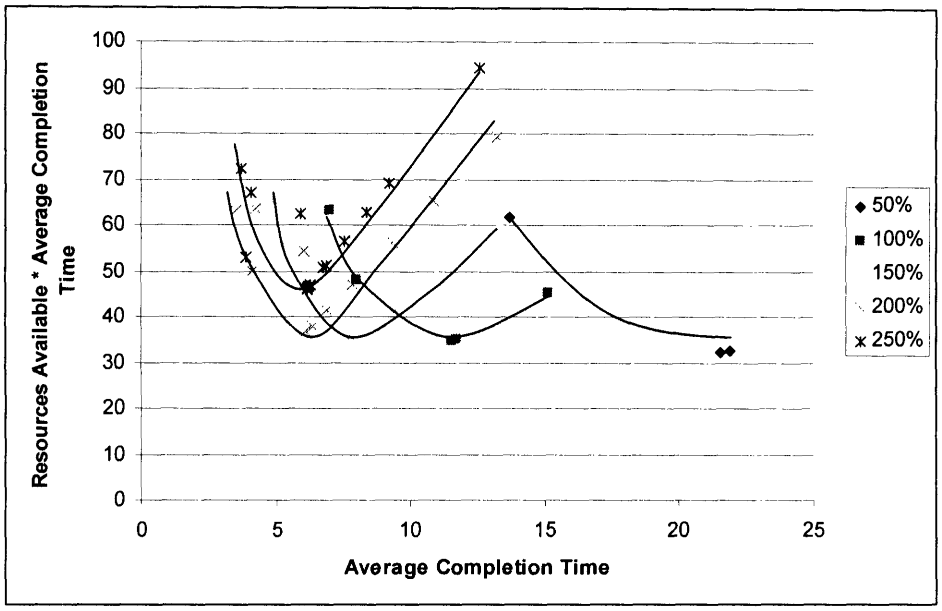


Figure 25: Master Simulation Chart for Example DSM

In the above figure, the different series represent the different starting values for the resources available. For example, the 50% series begins with a resource level of 0.5 for each type, and the utilization factor can only reach 0.5 before the constraint hits and resource relaxation begins. In contrast, the 250% series begins with a resource level of 2.5 for each type, and the factor ranges from 0.5 to 2.5 before the relaxation begins. As the series show, there are any number of resource and factor combinations that can produce a given completion time and cost. Also, any increase above 200% is useless because of the cap on the factor at 2 and the impossibility to improve past a 50% reduction in completion time. This restriction causes the 250% series to basically produce the same completion times as the 200% series, just at the higher costs of maintaining extra resources that aren't used. Overall, the point of the chart is that a frontier can be built for the DSM by considering the points furthest to the down and left. If connected, these points create the following curve:

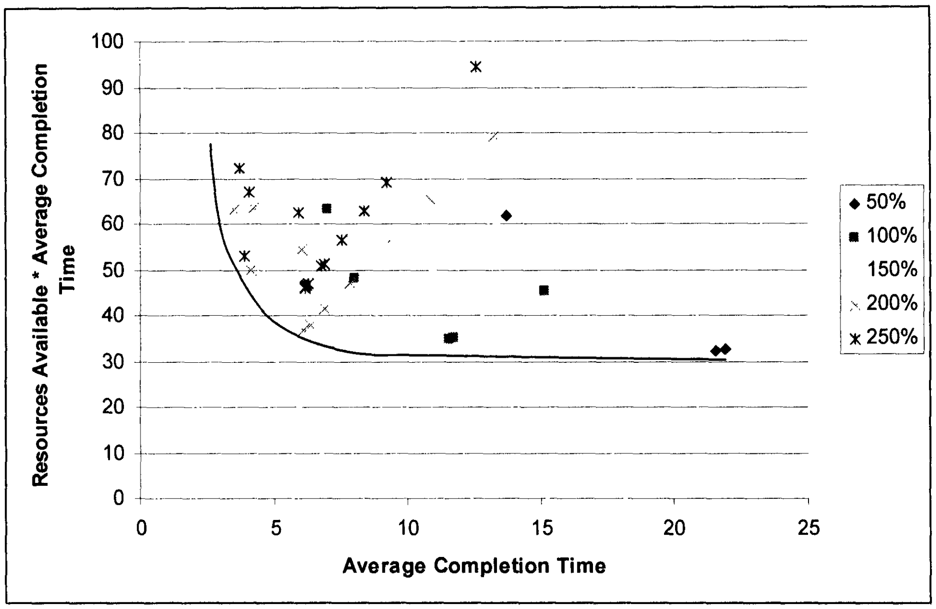


Figure 26: Production Frontier for the Example DSM

As the above figure shows, the frontier is bounded on the left by the resource relaxation curves as tasks reach their theoretical minimum times and on the bottom by the resource constraint positions. Obviously, choosing a point inside the frontier is sub-optimal, since reductions in either completion time or cost can be obtained by moving towards the frontier. Interestingly, the bottom of the frontier appears to be a horizontal line, implying that the effort to minimize cost can result in a wide range of completion times depending on the starting level of resources. So, if a company has a zero value for time (or a flat budget curve), they should pick the minimum completion time from the set of tangent points, which would be the point where the frontier begins to turn upwards. Thus, it is possible to truncate the frontier to just the curved portion, after which the budget curve can be used to pick the relevant tangent point. Presumably, the frontier would then shift down and to the left as the minimum completion times of the tasks are reduced through better information processing, technologies, databases, communication tools, etc. In addition, calculating the new curve can help determine the marginal benefit of investment in the acceleration of the development process.

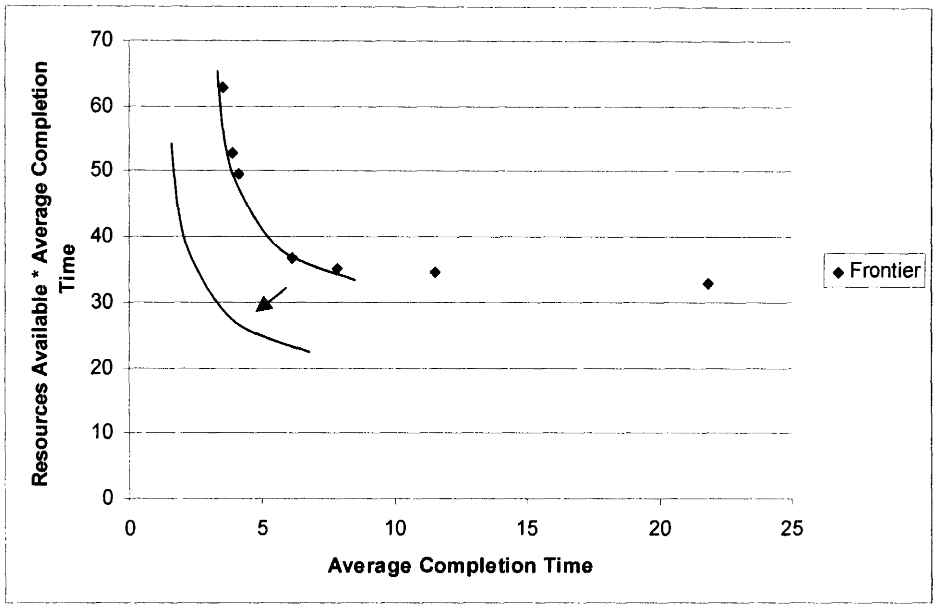


Figure 27: Shifting development frontier

Overall, these theoretical results have significant implications for managing a development process. First of all, it pays to use dedicate resources on single tasks as long as task times are reduced proportionally by the increase in resources. Essentially, this means that firms should be continuously pursuing efforts to reduce task times, either through the application of resources or more efficient use of those resources. If this action is taken, then the firm will always be on the frontier of its development process. Secondly, a budget constraint can be used to determine the optimal amount of resources to carry above the minimum. Significantly, this implies that resources should always be available in the development process, either to work on the next task in the sequence or to be ready for the next push in productivity. Unfortunately, companies are usually operating in a resource constrained environment, especially in a multiple program scenario.

Multiple Programs

Finally, the model was extended to multiple programs building off the resource additions. Essentially, the simulation tracks the work on separate programs (which are not required to be identical) as they draw resources from the same resource pool. The programs are offset by specified delays, and completion is reached when the work remaining on the last program is finished. Other than these additions, the model is the same as previously outlined. One notable consequence of this addition is that the delay between projects can be optimized for a given resource level. The goal is to find the delay that streamlines the even use of resources rather than combining peaks and valleys. In other words, the desire is destructive interference, not constructive interference. A sample simulation follows:

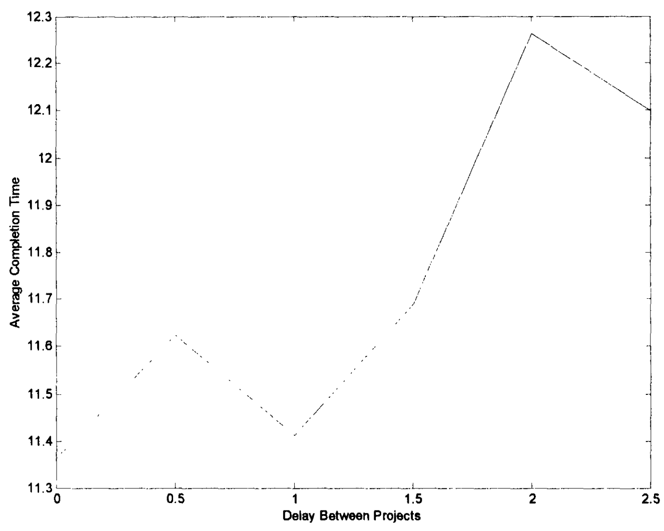


Figure 28: Simulation Results of Multiple Programs versus Delays

In the above figure, two identical projects were run with a resource constraint over a variety of delays. As the results show, the expected completion time for the combined projects was nearly identical for a delay of 0 or 1 day. However, a delay of 0.5 days would lengthen the overall completion time because of the pattern of resource use in the programs. In this manner, it's relatively easy to identify the optimum delay between projects for a given resource level.

Comments on Possible Further Extensions

Obviously, the model extensions are still fairly basic and do not include many of the more complicated items that affect resource allocation. Training time, errors, pre-planning, and systems engineering are only implicitly assumed, at best. However, the resource model does provide a method for optimizing the resource allocation based on a budget constraint. Notably, any budget constraint that values time above zero requires the project to maintain a resource level greater than the minimum required to do the project. Thus, the model is essentially telling companies to maintain a buffer of resources to work on more than one task at a time, or else they will be sacrificing performance. Presumably, these resources would need to be experienced engineers that have been on the project for some time so that they avoid the burdens of training, inadequate pre-planning and errors. In order to meet this level of experienced staff, the project has to been staffed much earlier that would normally be thought *(Repenning, 2001).*

Not surprisingly, it is better to have more people on a project early so that they can build experience and discover errors early in the process. Mathematically, this can be represented by decomposing the interaction values in the DSM to the sum of two values:

$$\textit{Interaction} = \textit{Base Interaction} + \textit{Error Rate}$$

Equation 7: Base and error rate decomposition

In this form, the 30% used as the example interaction value could be comprised of 25% base interaction probability with a 5% error rate across the entire system. If this error rate could be reduced, the total interaction value would fall and result in a lower likelihood of iterations for the entire matrix and a lower resultant completion time. In fact, the assumptions used thus far in the example DSM (i.e., the same value for all interactions) generate a straightforward non-linear relationship between the assumed interaction value and the resulting iterations for the example DSM. The exploding relationship is similar in form to Little's Law for queue time as the utilization approaches 100%:

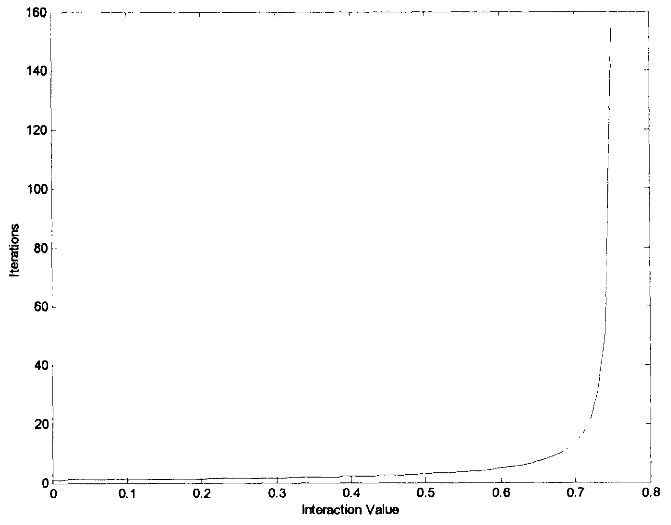
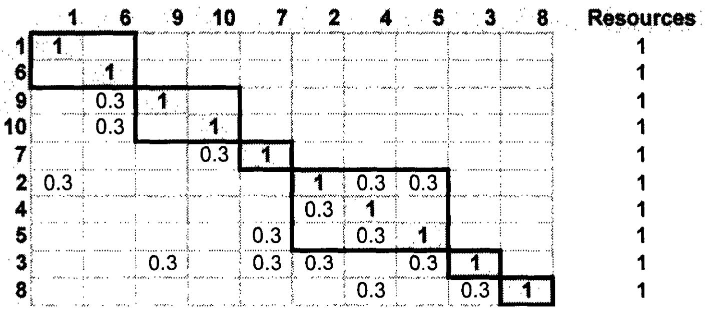


Figure 29: Non-linear relationship between iterations and interaction value

As the figure shows, reducing the interaction value from 30% to 25% by eliminating the errors will reduce the number of iterations from 1.64 to 1.5. Obviously, this benefits the convergence of the DSM and will result in a lower completion time. However, the improvement is highly dependent on the starting position of the DSM. In fact, the closer to non-convergence (or infinite iterations) that a system starts, the greater the benefit of reducing the error rate. Naturally, this should be a huge incentive to reduce the inherent error rate of the projects, or at the very least, create an incentive to discover errors early in the project. As Repenning showed in his system dynamics analysis, the best way to reduce these error rates and promote error discovery is to staff a project early with a large number of systems engineers dedicated to integration. This should also be combined with early staffing of the engineering team to promote early training and building experience on the project. One possible future extension is to incorporate error reduction methods into the simulation.

Reducing the DSM

In the course of dealing with larger DSM's, it became clear that reducing the DSM to a more manageable form would be useful. The primary assumption in this reduction will be that each identified block of tasks will be treated as one task and collapsed to equivalent values. This reduction needs to cover all of the variables in the matrix, including the durations, interaction values, and resource requirements. Obviously, as the DSM is reordered, the durations, interactions, and resources assigned to each task travel with the reordering. So, at the end of the new algorithm, the final example DSM looks as follows:



	1	6	9	10	7	2	4	5	3	8	Resources
1	1										1
6		1									1
9		0.3	1								1
10		0.3		1							1
7				0.3	1						1
2	0.3					1	0.3	0.3			1
4						0.3	1				1
5					0.3		0.3	1			1
3			0.3		0.3	0.3		0.3	1		1
8							0.3		0.3	1	1

Figure 30: Base resource configuration

In this particular example, 1's are used for both the task duration (in days) and the number of resources (in persons), thereby implying that each task takes 1 person-day to complete.

Interaction values outside of the blocks simply mean that the block is delivering information to another block. As the DSM is subsequently reduced, all of the variables need to be tracked appropriately. For the durations and interaction strengths, the reductions are relatively simple. New durations are calculated by simulating the blocks and replacing the set of tasks with the new duration and standard deviation. The only significant difference is that the data for a simulated block is now modeled using a lognormal distribution instead of the triangular distribution from the base simulation. The lognormal distribution is required to capture the effects of the long-tail distribution that a simple normal or triangular would miss. The lognormal distribution is also truncated at the minimum simulated time so that no zero durations are calculated. Otherwise, interaction strengths are reduced by taking the average of the non-zero items within the rows and columns.

However, the resource reduction takes a bit more work after determining the completion time. If the block were done in perfectly sequential order, the equivalent resource allocation (r') would be found by taking the sumproduct of the individual durations and resource requirements and dividing by the sum of the durations, essentially implying that one resource can bounce from task to task in order:

$$r' = \frac{\sum_i r_i t_i}{\sum_i t_i}$$

Equation 8: Sequential resource calculation

In contrast, a completely parallel time calculation would assume that all resources are working at the same time. Thus, the resource allocation would be found by taking the sumproduct of the individual durations and resource requirements and dividing by the maximum individual duration:

$$r' = \frac{\sum_i r_i t_i}{\max(t_i)}$$

Equation 9: Parallel resource calculation

Naturally, these resource allocations for purely sequential and purely parallel times mark the lower and upper bounds for the real resource allocation from the simulation. In the case of the simulation, the equivalent resource allocation will be found by dividing the sumproduct by the average simulated time:

$$r' = \frac{\sum_i r_i t_i}{simulation(t_i)}$$

Equation 10: Simulated resource calculation

However, there is a chance that this resource allocation could be below the largest required resource level for any given task, so this calculated resource level is compared to the requirements and the maximum value of the two is chosen:

$$r' = \max\left[\frac{\sum_i r_i t_i}{simulation(t_i)}, \max(r_i)\right]$$

Equation 11: Improved simulated resource calculation

With this technique, the number of resources required for a particular set of tasks can be tracked as the DSM is reduced to a more viewable form. For the example matrix, the reduction is below:

	1'	9'	7	2'	3	8	Resources
1'	1						2
9'	0.3	1					2
7		0.3	1				1
2'	0.3		0.3	3.8			1
3		0.3	0.3	0.3	1		1
8				0.3	0.3	1	1

Figure 31: Reduced DSM with simulated times and resources

As this new matrix shows, both the completion time and number of person-days in the reduced DSM is higher than in the original representation shown in Figure 17. This new matrix has 10.8 person-days of work because the iterative block 2' has been reduced to one representative task, and the completion time is 8.8 days. These differences are due to the blocking of the matrix, which essentially forces a block to obey the most restrictive set of rules of its component tasks. In this particular case, the original simulation allows task 2 to proceed after its single upstream task 1 is complete. However, blocking and reducing the matrix essentially makes task 7 an upstream precedent for task 2, since task 7 is a precedent for task 5 and the assumption is that tasks 2, 4, and 5 are going to be worked together. Consequently, task 2 must now wait and this waiting drives up the overall completion time.

Overall, this increase in completion time is a result of the tightly coupled tasks inside a block being executed sequentially in the simulation. However, this calculation is open to discussion because of the assumption of sequential execution. If a manager were unaware of the relationships between the three tasks (2, 4, and 5), then the sequential assumption is probably valid – the tasks will proceed in order and generate rework as appropriate. However, a priori knowledge of the relationships could prompt the manager to execute the tasks in a purely parallel arrangement, essentially by throwing everyone into a conference room and getting them to work jointly on the three tasks. If this purely parallel assumption is followed, then the reduced matrix is the following:

	1'	9'	7	2'	3	8	Resources
1'	1						2
9'	0.3	1					2
7		0.3	1				1
2'	0.3		0.3	1.9			3
3		0.3	0.3	0.3	1		1
8				0.3	0.3	1	1

Figure 32: Reduced Matrix with Parallel Times and Resources

In this version of the reduced matrix, the completion time is 6.9, and the number of person-days is 12.7. Also notice that the resource allocation follows the convention for the purely parallel time since the assumption is that everyone is working in each time period. Conveniently, this second method of calculating provides another illustration of the trade-off between completion time and cost. The extra 1.9 resource-days are essentially used to buy the 1.9 days reduction in completion time. Although this calculation is interesting, it's unlikely that managers will be able to approach the lower bound of the completion time for a given set of tasks, even if they know the a priori relationships. For this reason, the model will continue to use the simulation times for the reduction and the additional time penalty will be included as part of the trade-off for additional clarity in blocking and gating the development process.

Throughout the process of reordering and collapsing the core DSM matrix, the SD, impact, and probability matrices were processed similarly. Standard rules were used to combine the individual deviations by summing variances where applicable, and simulation was used to determine the new deviation when a direct calculation was not possible. For the example matrix, the final SD matrix became the following:

	1'	9'	7	2'	3	8
1'	0.25	0	0	0	0	0
9'	0.05	0.25	0	0	0	0
7	0	0.05	0.25	0	0	0
2'	0.05	0	0.05	0.87	0	0
3	0	0.05	0.05	0.05	0.25	0
8	0	0	0	0.05	0.05	0.25

Figure 33: Final collapsed SD matrix for example DSM

Finally, it's important to recognize that the reduced DSM's are completely sequential. As such, the simulation will generate results that are a straightforward sum of the durations since all of the iterations have been taken into account by considering each block of tasks separately. This sequential nature hampers many of the previous resource calculations, mostly because tasks are not being run in parallel anymore. However, the resource calculations will be useful again for combining multiple projects that pull from the same resource pool.

Eigenvalue Tearing

In general, there are three methods for reducing the completion and convergence time for a DSM in addition to reordering and adjusting resource levels:

1. Reduce the individual task times
2. Reduce the number of iterations
3. Eliminate tasks

All policy and tool recommendations for any project or process will fall into one of these three categories. First of all, enhanced databases, real-time information, and communication tools are merely a few examples of tools designed to reduce task times. They generally allow the same number of iterations to be performed faster, thereby reducing the overall development time. Secondly, standards and company policies are often designed to eliminate interactions and reduce the number of required iterations for convergence. A great example is the use of a standard car hinge for all car doors and a standard truck hinge for all truck doors, as opposed to developing a new hinge for each program. Thirdly, elimination of tasks through policies or standards is one of the most powerful options to improve performance.

In general, the second option is a more effective and permanent way to improve a process and the best way to prioritize the reduction of iterations is through eigenvalue tearing. The eigenvalues of the matrix control the rate of convergence, and the tearing process examines each interaction to figure out which ones have the greatest effect when removed. As discussed in *Smith and Eppinger*, the eigenvalues and eigenvectors of the DSM can be used to identify the dominant modes of work. The eigenvalues represent the geometric rate of convergence of the given mode, with a real value less than 1 representing convergence, a real value greater than 1 representing divergence, and a complex value representing oscillation that can diverge or converge as well. Eigenvectors contain information on the relative importance of each task in a given design mode, allowing us to identify the dominant sub-problems within each parallel block. The interpretation of eigenvalues and eigenvectors in an organizational context is very similar to a physical system such as electronics, biology, chemistry, or geology. Because of the coupled non-negative structure of the DSM, the largest magnitude eigenvalue must be real and positive *(Smith and Eppinger, March 1997)*. By extension, the eigenvector associated with this eigenvalue will also have positive elements. Therefore, the slowest design mode, which corresponds to the largest eigenvalue, will be relatively easy to interpret. It is not certain that the other design modes will be so simple to interpret, as they may include complex numbers and oscillating behavior. Regardless, the overall behavior of the process convergence will be determined by the largest eigenvalue, so the analytical focus is best reserved for that area.

Following the example matrix, the base example DSM has an eigenvalue of 0.39. A rough approximation for the upper bound on the number of iterations to completion comes from the following formula:

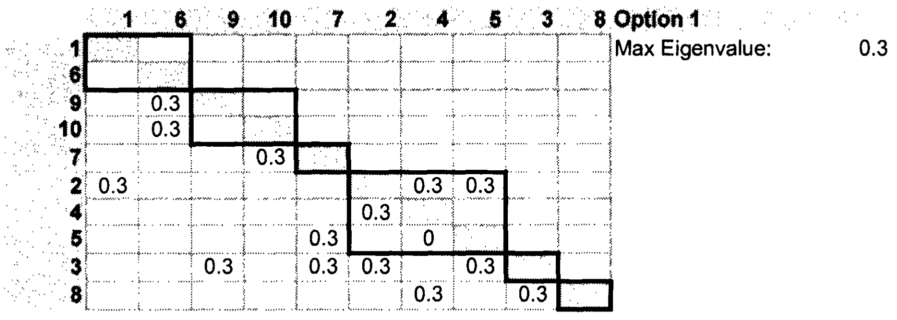
$$iterations = \frac{1}{1-\lambda}, \lambda < 1$$

Equation 12: Controlling limit of iterations

where λ is the maximum real eigenvalue. In the example, this formula suggests that the process will have to iterate 1/(1-.39) = 1.64 times before converging. As far as DSM's go, this is a relatively small number. Initial cuts of the base DSM at Magma suggest a maximum eigenvalue in the range of 0.8-0.9, generating 5 to 10 iterations before a design converges. Since no engineer reasonably expects to perform his same job 5 to 10 times as part of a repeating loop before declaring it complete, it is highly likely non-converged designs are passed on.

In light of the importance of the eigenvalue and its dependence on the interaction values within the DSM, the tearing process is designed to systematically examine all interactions for their contribution to the maximum eigenvalue. In general, this process is conducted in reverse by removing an interaction above the diagonal from the DSM and calculating the new maximum eigenvalue. The largest change is kept, the DSM is reordered if necessary, and then the process repeats until a threshold value is reached. In this manner, all of the controlling interactions can be identified and removed until a given target is reached.

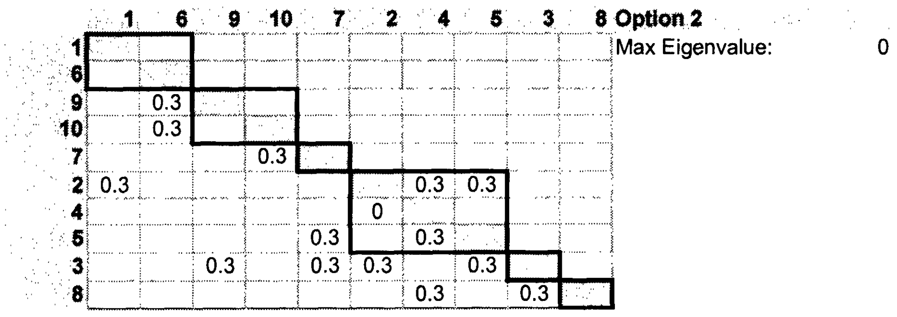
For the example matrix, the two possible scenarios as shown below:



	1	6	9	10	7	2	4	5	3	8
1										
6										
9		0.3								
10		0.3								
7				0.3						
2	0.3						0.3	0.3		
4						0.3				
5					0.3		0			
3			0.3		0.3	0.3		0.3		
8							0.3		0.3	

Option 1

Max Eigenvalue: 0.3



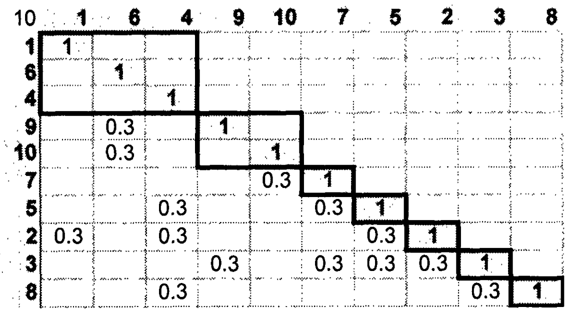
	1	6	9	10	7	2	4	5	3	8
1										
6										
9		0.3								
10		0.3								
7				0.3						
2	0.3						0.3	0.3		
4						0				
5					0.3		0.3			
3			0.3		0.3	0.3		0.3		
8							0.3		0.3	

Option 2

Max Eigenvalue: 0

Figure 34: Examples of eigenvalue tearing

where the gray zeros mark the interactions that were removed. As can be seen, the second option is preferable over the first because the second eliminates all precedents from task #4 and creates a perfectly sequential process. Whereas the first option still requires 1.43 iterations to converge, the second requires only 1. In fact, reordering the second matrix would produce the following:



10	1	6	4	9	10	7	5	2	3	8
1	1									
6		1								
4			1							
9		0.3		1						
10		0.3			1					
7					0.3	1				
5			0.3			0.3	1			
2	0.3		0.3				0.3	1		
3				0.3		0.3	0.3	0.3	1	
8			0.3						0.3	1

Figure 35: Final torn and reordered matrix

Clearly, this torn matrix is vastly preferable to the previous one because all iterations are removed. The process must only be completed once to reach a converged design, and it requires 7 days instead of the simulated 8.8 days for the original blocked example in Figure 31. However, the completion time is actually slightly slower than the version using a parallel time calculation in Figure 32. Forcing tasks #5 and #2 to be sequential adds a fraction of time to increase the total to 7 days from the previous 6.9 days. However, this trade-off is usually preferable for the additional certainty in the final product. In fact, this trade-off between completion time and certainty can be explored further.

As a DSM is torn apart, the removal of the interactions breaks apart coupled blocks into smaller and smaller pieces. As a general rule, de-coupling the iterative blocks will decrease the eigenvalue of the system and result in a smaller number of iterations and a smaller completion time. However, when the removal of an interaction creates the opportunity for reordering and a separation of one block into two, thereby allowing the completion time to actually increase since the first block must be completed before the second block starts. If the blocking were not considered and the full simulation were run, then the time would universally decrease. Taken to the extreme, the removal of all interactions and retaining blocking will reduce the system to a simple sequential order of tasks. Graphically, it looks as follows for the example matrix as interactions are torn out:

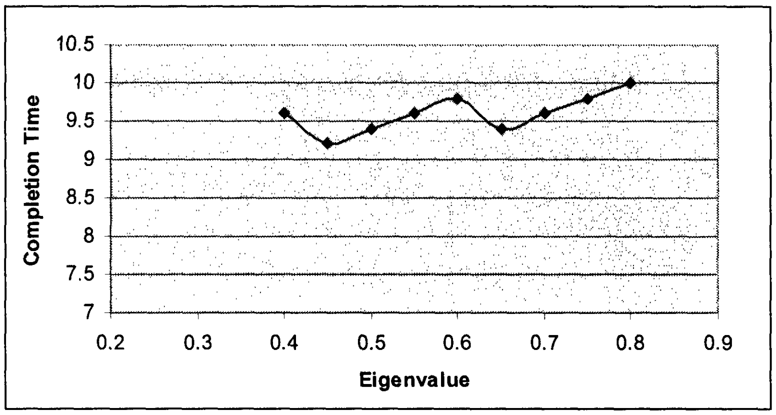


Figure 36: Variation of completion time as eigenvalue decreases

Regardless of this variation in the expected completion time, the standard deviation of the time and the sensitivity to changes in interaction strength falls continuously. This makes for a non-intuitive solution, as there may be cases where increased certainty is preferred over a lower average. For this reason, the 0.6 eigenvalue in the above chart may be better than the 0.65 eigenvalue, in spite of the higher average duration.

After tearing has been used to find the controlling interaction(s), the primary challenge is determining the standard or policy that will eliminate the interaction or reduce it to zero time. Often, a creative solution can be found by talking to the parties involved in the primary design mode that is associated with the eigenvalue. For the example DSM, the design mode reveals that the most work is being done by tasks #5, 4, 2, 3, and 8. Examining these tasks together with the people involved would likely reveal some interesting insights into the difficulties and sources of iteration.

Advanced Eigenvalue Tearing

In the course of developing the complete dataset for Magma's closures process, it became clear that the traditional method of eigenvalue tearing would be inadequate. Quite simply, the initial central coupled block of the process included 450 tasks, which is far too large to check every interaction for the best one to remove. As a result, some amount of prioritization was required to speed the computation of the tearing process.

From example matrices and practice tears, it was clear that the best interactions to remove were always related to the tasks being iterated the most (as shown by the eigenvector). So, rather than test every interaction, it made sense to first identify the tasks working the most and then pursue their precedent and dependent interactions as the best tearing options. Naturally, these tasks were identified by calculating the eigenvalue and eigenvector matrices and examining the results. Ultimately, the new algorithm for tearing the re-ordered DSM became:

1. Identify maximum real eigenvalue and the accompanying eigenvector
2. Find maximum value in eigenvector and the associated task
3. Identify set of one-step interactions related to chosen task above the diagonal
4. Find best single interaction to remove and remove it
5. Check for reordering of DSM
6. Go to step 1

Steps 1-4 are fairly self-explanatory because they are used to process the behavior of the DSM and identify the most likely places for an interaction to be removed. It's not surprising that the best interactions to remove involve the tasks being iterated the most. However, it was not initially obvious that the DSM would need to be reordered after each removal. In retrospect, the reordering is required to remove tasks that are no longer part of the core coupled block. If this is not done, then the original task will still show up as a highly worked task and mislead the tearing process. In practice, the operation looks as follows for the same sub-matrix torn above:

	1	2	3
1		0.3	
2			0.3
3	0.3	0.3	

Figure 37: Example DSM matrix for advanced tearing

Eigenvalue	1	2	3
1	0.3974	0	0
2	0	-0.1987 + 0.1687i	0
3	0	0	-0.1987 - 0.1687i

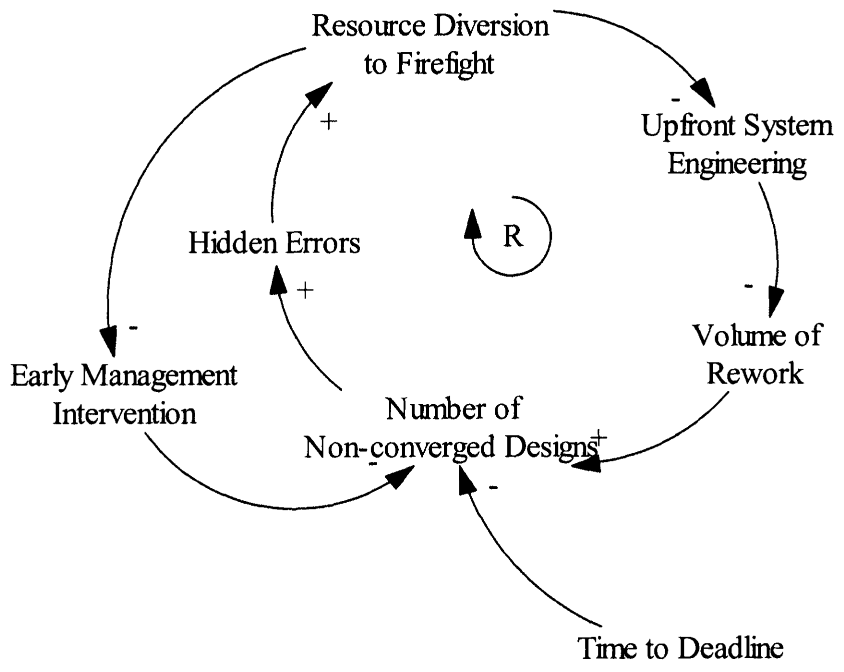
Eigenvector	1	2	3
1	-0.414	0.6559	0.6559
2	-0.5484	-0.4344 + 0.3688i	-0.4344 - 0.3688i
3	-0.7265	0.0804 - 0.4885i	0.0804 + 0.4885i

Figure 38: Steps 1-2 of advanced tearing process

From the above tables, it's clear that the maximum real eigenvalue is 0.39, and that the associated eigenvector is in column 1. In the eigenvector, the task performing the most work is #3, suggesting that the interactions up for consideration are 3-1, 3-2, and 2-3. Of this set, only 2-3 is above the diagonal, so it is the best choice for removal. Individually checking the removal of each listed interaction would also produce the same results, since the new eigenvalues would be 0.3, 0.3, and 0, respectively.

III. Magma Analysis

As presented in the introduction and context, Magma wants to focus on improving its closures development process, which is mired in a traditional project management negative loop. Currently, high levels of hidden errors are present in the system, as evidenced by the large amount of work required of the pre-production launch teams for each program. This forces the best engineers and management to be diverted to launch teams, where their input and expertise is rightfully valued. However, their presence on the launch teams precludes their involvement on the systems portion of programs in the beginning stages, thereby creating the hidden errors that they will be forced to solve at the last minute on the next launch. The following figure generically illustrates the reinforcing loop:



Resource Diversion to Firefight
-
Upfront System Engineering
-
Volume of Rework
+
Number of Non-converged Designs
+
Hidden Errors
+
R
-
Early Management Intervention
-
-
Time to Deadline

Figure 39: Firefighting reinforcing loop

From this simple diagram, it is simple to see several methods to escape the loop. First of all, deadlines can be extended to reduce the number of non-converged designs. Secondly, extra resources can be added to both firefight and do upfront engineering at the same time. Finally, DSM's or other reengineering tools can be used to speed up the development time so that the rework is completed before the deadline. However, Magma has been reluctant to slip deadlines (which damage the product timeline), add resources (which are expensive), or implement the changes from prior DSM analysis (which requires dramatic changes to decades old processes). Consequently, Magma has managed to address the same set of problems (i.e., water leaks, wind noise, NVH, etc.) over decades for successive programs without ever resolving the root causes. In some respects, the use of surrogate vehicles has enhanced this problem, since those surrogates inevitably contain the very same problems that Magma is trying to solve in the next generation vehicle. Ultimately, Magma has a development process that does not naturally converge in a closed fashion, but rather requires significant management interference to produce a product at the end of the day. Unfortunately, this entire vehicle development process is much too large to analyze in a short time frame. For that reason, the closures process (doors, trunks, and hoods) was chosen to be a meaningful and significant, yet tractable, piece of analysis.

Data Sources & Methods

Fortunately, Magma has a massive amount of process data that can be co-opted for project management analysis. As a basis for their vehicle development process, Magma uses the Magma Product Development System (MPDS) as a general guideline for activities, milestones, and requirements. MPDS is primarily designed as a massive timeline, covering up to ~4 years before the launch of Job #1 off the assembly line. The basic outline of the MPDS process follows:

	Kick-off	Pre-Strategic Intent	Strategic Intent	Strategic Confirmation	Proportions & Hardpoints	Program Approval	Surface Transfer	Product Readiness	Confirmation Prototype	Change Cut-off	Launch Readiness	Launch Sign-off	Job #1
Abbreviation	KO	PS	SI	SC	PH	PA	ST	PR	CP	CC	LR	LS	J1
New Vehicle	0.0%	5.8%	19.2%	30.8%	35.6%	42.3%	51.0%	63.5%	72.1%	84.6%	91.3%	93.8%	100.0%
New Exterior, Modified Structure	0.0%	6.0%	18.0%	28.0%	33.0%	40.0%	49.0%	62.0%	71.0%	84.0%	91.0%	93.5%	100.0%
New Exterior	0.0%	7.0%	14.0%	20.9%	25.6%	33.7%	41.9%	57.0%	67.4%	81.4%	89.5%	92.4%	100.0%
Moderate Freshening	0.0%	7.9%	15.8%			34.2%	42.1%	52.6%	63.2%	78.9%	88.2%		100.0%
Minor Fresheing	0.0%		14.3%			28.6%	39.3%	53.6%	64.3%	78.6%	87.5%		100.0%
Trim	0.0%		14.3%			23.8%	33.3%	47.6%	61.9%	76.2%	83.3%		100.0%
	Design Work (Area of Focus)								Troubleshooting, Rework, and Final Changes				

Figure 40: MPDS outline of timing and scalability

where the percentages in the table represent the progression towards Job #1 in each phase. In reality, not all of the MPDS process is new engineering work. In fact, the engineering and product design Job #1 occurs at product readiness (<PR>), with the troubleshooting and rework beginning shortly thereafter. If the system were running in perfect fashion to produce converged designs at <PR>, then the confirmation prototypes produced in the next phase should be just that: confirmation of the current design. Unfortunately, the confirmation prototypes are often where many of the initial mismatches and mistakes are discovered, prompting a significant amount of rework. By itself, this official pattern suggests that Magma is certainly not generating converged designs at <PR> from the product development side, much less the combination of product development and manufacturing. Nevertheless, this realization allows me to restrict my analysis to the time frame before <PR>, where the tasks producing the final design are executed.

In addition to the MPDS, the major sources of data were:

1. Feasibility Checkpoint (FC) process
2. Magma Manufacturing Development System (MMDS)
3. Manufacturing Design Specifications (MDS)
4. Other Interactions (interviews and common sense)

By far, the most useful piece of data was the FC process. As part of his work at Magma, Tony Zambito performed a DSM analysis on the product development process for the entire vehicle. This analysis generated a DSM of 920 tasks, complete with individual task durations. However, interaction strength was not captured in the analysis. From this base DSM, Zambito was able to re-order the MPDS tasks and generate significant parallel iterative blocks. At the end of each of these blocks, a feasibility checkpoint (FC) was created to ensure that one block was complete before the next began. In this manner, five major checkpoints were added to the MPDS process. Unfortunately, the FC results were also forced into the standard timeline format by adding extra dependencies, thereby eliminating much of the iterative information discovered in the original analysis.

The MMDS process is a set of work designed to capture the entire manufacturing development process. In the same way that the product design engineers are generating a design, the manufacturing engineers are generating a process design to match the product. The MMDS is a flowchart representation of the manufacturing design process, showing the progress along the MPDS timeline. A sample portion of the MMDS is below:

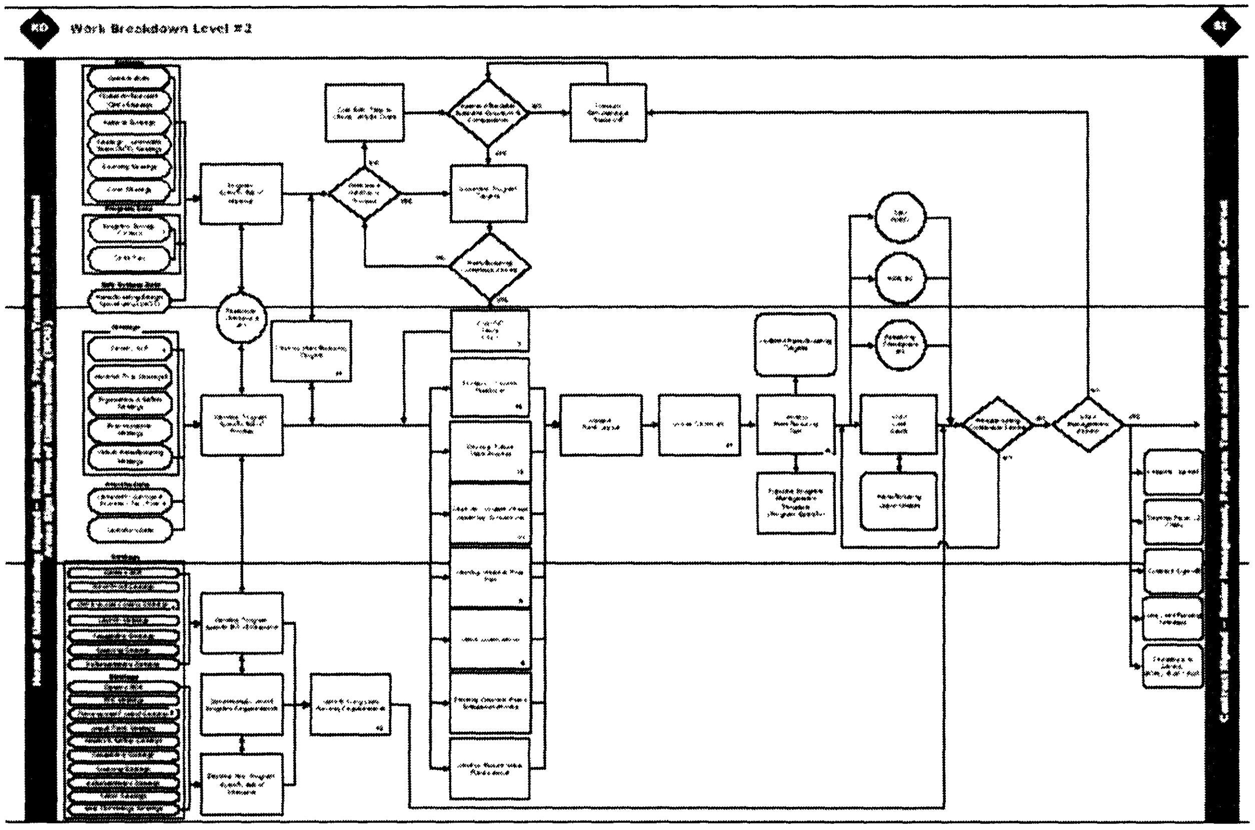


Figure 41: Sample portion of the MMDS: <KO> to <SI>

As is shown, the flowchart is quite convenient for establishing precedents and dependents within the MMDS process.

By themselves, the FC process and MMDS process gave me all of the tasks required to complete the product and process development for closures. The closures related tasks were extracted from the overall FC process by Todd Dishman as part of a system integrators project, and the entire MMDS process was used because of its general nature. Essentially, the MMDS describes the process of developing an assembly line and is universally applicable to any part of the vehicle development process. The FC and MMDS data also allowed me to initially fill in a large portion of the interactions in the final DSM. Graphically, the data up to this point looks as follows:

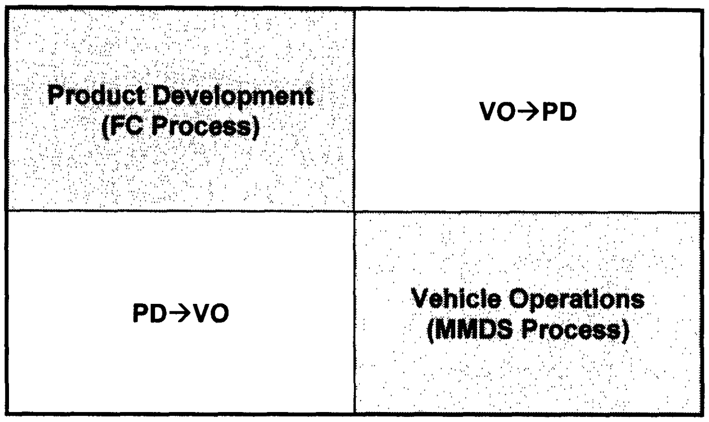

Product Development (FC Process)
VO→PD
PD→VO
Vehicle Operations (MMDS Process)

Figure 42: Initial closures DSM from FC and MMDS data

As can be seen from the figure, each set of data makes up part of overall closures DSM. However, nothing is currently in either the upper right or lower left boxes of the DSM, which essentially means that none of the major interactions from PD to VO or vice versa are being represented. Some of these interactions can be filled in by examining the MMDS process closely, for part of the flowchart lists "external" inputs that come from the FC process. These external inputs tend to be one-way interactions, since PD is flowing specifications and inputs to VO, or other inputs are coming in from the outside. In addition, there are also a few tasks that are exactly duplicated between the two data sets, allowing their interactions to be summed and one task to be eliminated. A sample duplicate is the Global Architecture Strategy, which is essentially a completely outside input to both the product development and manufacturing organizations. However, the remaining interactions must be gleaned from the last two data sources.

The Manufacturing Design Specifications (MDS) are set up to be a direct feed of information from the Vehicle Operations group back to Product Development. By design, there are certain specifications that product development activities must meet before a design can be approved. In general, most of these MDS's are being combined with another set of specifications that PD must follow, but the combination isn't complete yet. As a result, the MDS was used to fill in a portion of the VO→PD box in the overall DSM. These interactions are distinctly one-way interactions, since an MDS specification flows only from VO to PD, and PD cannot change an MDS. Finally, the last data source is simply a set of interviews that generated a list of likely interactions between VO and PD, covering a range of tasks and activities. In general, these interactions covered two-way iterative loops, where PD could affect VO as much as VO could affect PD. An example is the product and process feasibility check in the MMDS process, which verifies that the product design can actually be produced in the factory. If the product cannot, VO feeds information back to PD, which often has to change the clay models and base design of the vehicle. Obviously, this new clay model then feeds back into the product and process feasibility for a reevaluation. As such, these interactions have to be mapped both to the upper right and lower left of the new DSM. Graphically, the complete data set looks as follows:

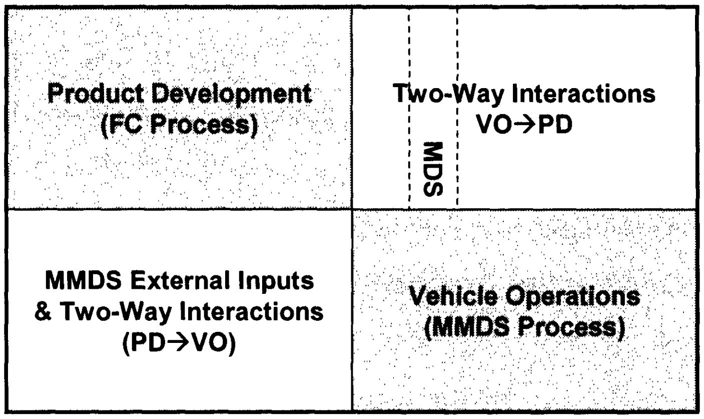

Product Development (FC Process)
Two-Way Interactions VO→PD
MDS
MMDS External Inputs & Two-Way Interactions (PD→VO)
Vehicle Operations (MMDS Process)

Figure 43: Complete closures DSM

At this point, all of the tasks and the interactions have been included for the closures DSM. However, the overall strengths of the interaction and the duration of the MMDS tasks still remain unknown.

Data Estimation

Task Duration

For the vast majority of tasks, durations came from the original FC process. For those tasks missing duration values, the FC tasks were used as guidelines to estimate durations before checking those durations with more experienced people.

Strength of Interaction

Typically, DSM analyses ask executors of tasks to rank their dependency on other tasks on a scale of 1-3 or perhaps 1-9. These ratings are then mapped to a range of interaction probabilities, typically 10%-50%. The DSM can then be analyzed for completion times and number of iterations, based on the inherent Markov chain represented within the matrix. Unfortunately, the initial creation of the DSM for the FC database did not capture any strength of interaction metrics.

As a result, I had to determine another method for estimating the strength of interaction for the 773-by-773 closures matrix. Because of the matrix size, repeating interviews was simply not a viable option in the allotted time. Rather, an examination of the sensitivity of the timing calculations had to be used to establish a likely range for the strength. Fortunately, a couple of assumptions and clues exist to help narrow the range. Although the proprietary data for the current process must be protected, it is widely known in the automotive world that design processes generally take about 4 years to complete because of a standard product cycle. With ~250 working days in each calendar year, this implies that a generic target of 1,000 days is useful for reference purposes during the analysis. Furthermore, this generic target of 1,000 days will be taken as the desired maximum time for convergence of the design process. However, I know that the current process is probably not converged because of the large number of errors that are fixed during the vehicle launch. Thus, some iteration is being skipped in the actual process, and the theoretical time must be greater than 1,000 days. Second, I assume that the parallel time to execute the tasks is shorter than the sequential time, owing to the fact that processing tasks in parallel should be faster than in series. Of course, I know this result won't always be true as the DSM approaches an eigenvalue of 1, but this switch will prove useful in helping me identify the appropriate range of behavior. Third, I created a current task order as best as possible, by ordering the tasks according to their start date listed in the original FC database. Finally, I have to assume that every interaction has the same strength for simplicity of analysis.

In order to begin the examination, a graph of the purely sequential and purely parallel times was produced by varying the interaction value uniformly across the matrix:

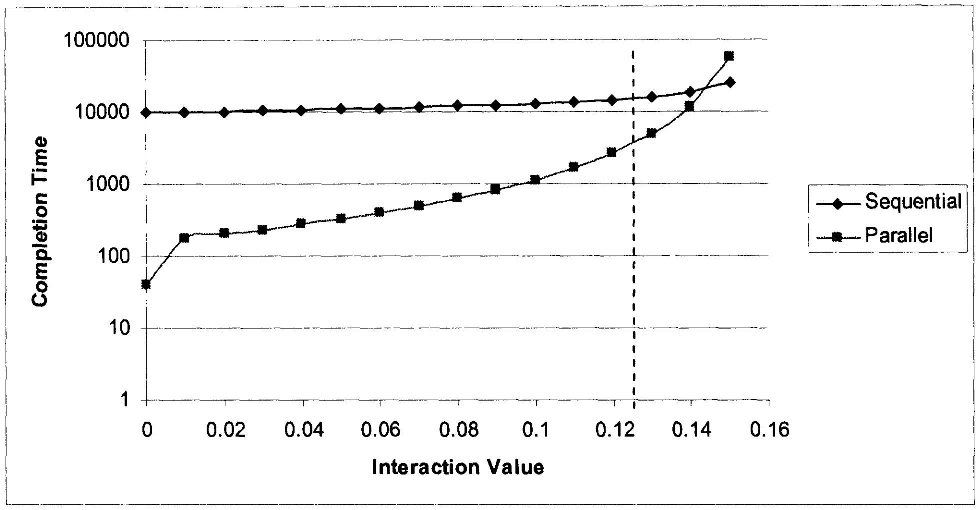


Figure 44: Timing sensitivity to strength of interaction (as-is order)

As the above chart shows, the parallel and sequential curves are both above 1,000 days after interaction strength of 10%. However, they cross at 14.5% as the parallel timing calculation shoots off to infinity as the eigenvalue approaches 1. As a consequence, the timing calculations suggest that the appropriate range for interaction strength is somewhere between 12% and 13% mainly because this range fits the a priori assumptions about the process. However, a reality check clearly indicates that this value is somewhat low, at least for individual interaction values on the level of tasks that are included in this DSM. For example, changing the plant layout probably has a greater expected value of interaction with the assembly order than 12.5%. However, the use of a generic interaction value as an average does tend to produce different results than actually simulating the combination of varying values. Since it is impossible to obtain the actual values in the allotted time frame, the analysis will be conducted with a generic, equivalent, value for all interactions. An expanded view on the sensitivity of this assumption follows in the plot of the eigenvalue against the generic interaction strength:

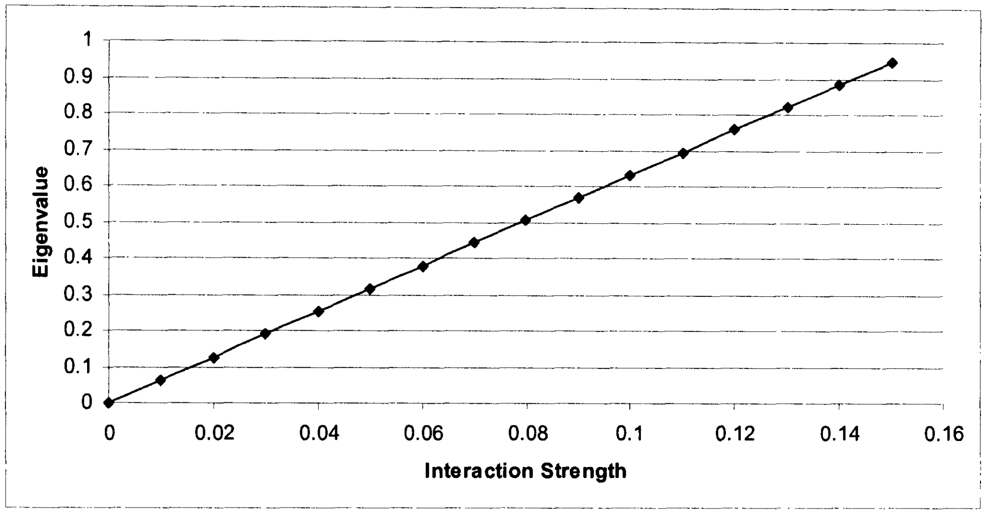


Figure 45: Plot of eigenvalue vs. interaction strength for closures DSM

As an extrapolation of the chart shows, a more realistic assumption of 25% for the interaction strength would generate an eigenvalue of ~1.6 for the closures DSM. This would imply that the product and process development would *never* converge to a solution in any amount of time if the expected value of 25% is the true value across the entire matrix. Obviously, this revelation has huge implications for the organization. Most significantly, this means that Magma's development process cannot produce a converged design by itself, as long as the DSM is capturing the true reasons for non-convergence. In other words, there will always be significant launch issues that affect basic system interactions and customer requirements. Thus, significant management involvement will be needed to force convergence by the deadline. As we will see, one set of solutions to this predicament is massive standardization of architectures, elimination of tasks, and improved analytical tools. Alternatively, dramatically increasing manufacturing flexibility or creating more consistent specifications can help to reduce the number of iterations.

DSM Analysis

Before processing the DSM, an interaction value had to be assumed for calculation purposes. After examining the eigenvalues associated with a 25% assumption, it seems possible that Magma is maintaining a non-converging process. However, processing the DSM would be impossible with an eigenvalue greater than 1. For this reasons, all of the following calculations were performed with a base interaction value of 12.5% to model the overall process, as developed above. Only at the end will the scenario including a 25% interaction value be revisited.

However, simply settling on the expected value of interaction was not sufficient to begin calculations using the simulation. Rather, the expected value of 12.5% needs to be broken out into a probability and impact value because the probabilities act as a gate in the simulation. In fact, an equal expected value of 12.5% will produce two different completion times if the probability and impact are 12.5% and 100% versus 25% and 50%, respectively. The method for estimating the split between probability and impact relied on the existence of the lower and upper bounds from the parallel and signal flow calculations. Running the simulation to match these upper and lower bounds through trial and error produced the following table:

	Lower Bound	Upper Bound
Time (days)	3,200	15,400
Probability	19.0%	22.0%
Impact	65.8%	56.8%
Expected Value	12.5%	12.5%

Figure 46: Bounding table for probabilities and impacts

As the table shows, the probability and impact values that produce the lower and upper bounds are not substantially different from each other. In addition, they both correspond to a rework impact value of ~60%, which makes intuitive sense. Reworking 100% of a task after an interaction is unrealistic, but so is 10%. Because of the relatively narrow range of the two values, choosing a middle value seems to be safe. Thus, the Magma calculations are being done with a probability of 20% and an impact of 62.5% for each interaction.

This assumption essentially gives Magma a converging process that takes between 3,200 (parallel) and 15,400 (sequential) days to reach completion (from Figure 44). Running the as-is order through the simulation generates an average time of 6,500 days, a standard deviation of 4,300 days, and a minimum of 3,200 days (lognormal distribution).

Reordering

Reordering the base closures DSM using the algorithm described above produced the following reduced matrix in which each single task represents a block of tasks from the original matrix:

DSM	18	1	38	37	36	279	278	311	4	565	564	259
18	40	0	0	0	0	0	0	0	0	0	0	0
1	0.125	30	0	0	0	0	0	0	0	0	0	0
38	0.125	0.125	30	0	0	0	0	0	0	0	0	0
37	0	0.125	0.125	20	0	0	0	0	0	0	0	0
36	0	0	0.125	0.125	15	0	0	0	0	0	0	0
279	0.125	0.125	0	0.125	0.125	20	0	0	0	0	0	0
278	0	0.125	0	0.125	0	0.125	20	0	0	0	0	0
311	0	0.125	0	0	0	0	0.125	15	0	0	0	0
4	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	7.668	0	0	0
565	0	0	0	0	0	0	0	0	0.125	15	0	0
564	0.125	0	0	0	0	0	0	0	0.125	0.125	30	0
259	0.125	0.125	0	0	0	0	0	0	0.125	0	0.125	20

Figure 47: Base re-ordered & reduced closures DSM

As the level of reduction suggests, the base DSM is actually a highly coupled set of interactions. Each parallel block was reduced to a single task, implying that all 773 tasks were reduced to a set of 12 blocks. This level of coupling is incredibly high, and is dominated by a central block of 391 tasks represented by task #4 above. The matrix of standard deviations is shown below:

SD	18	1	38	37	36	279	278	311	4	565	564	259
18	10	0	0	0	0	0	0	0	0	0	0	0
1	0.01875	7.5	0	0	0	0	0	0	0	0	0	0
38	0.01875	0.01875	7.5	0	0	0	0	0	0	0	0	0
37	0	0.01875	0.01875	5	0	0	0	0	0	0	0	0
36	0	0	0.01875	0.01875	3.75	0	0	0	0	0	0	0
279	0.01875	0.01875	0	0.01875	0.01875	5	0	0	0	0	0	0
278	0	0.01875	0	0.01875	0	0.01875	5	0	0	0	0	0
311	0	0.01875	0	0	0	0	0.01875	3.75	0	0	0	0
4	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	1.085+3535i	0	0	0
565	0	0	0	0	0	0	0	0	0.01875	3.75	0	0
564	0.01875	0	0	0	0	0	0	0	0.01875	0.01875	7.5	0
259	0.01875	0.01875	0	0	0	0	0	0	0.01875	0	0.01875	5

Figure 48: Standard deviation of re-ordered & reduced closures DSM

As a note to the calculation methods, the complex number in the SD matrix is used to signify a lognormal distribution and to store the two parameters required (1.085 + 3535i). The real portion of the complex is the standard deviation of the lognormal, and the imaginary portion is the minimum of the distribution from the simulation. These two parameters, when paired with the average of the lognormal in the DSM matrix (7.668), allow the retention of the lognormal behavior in the reduced DSM. When the DSM is randomized using the average and standard deviation matrices, the following truncated lognormal distribution is generated for 1,000 runs:

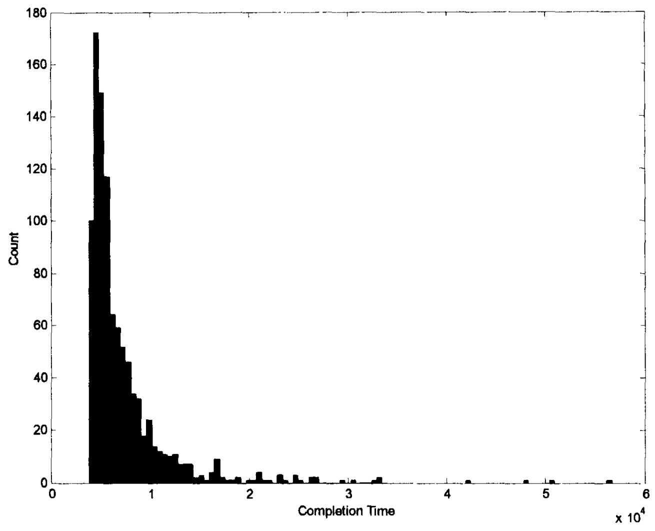


Figure 49: Histogram of reduced closures DSM completion times

The statistical result of this distribution is an average completion time of 7,345 days with a standard deviation of 4,815 days. Notably, these numbers are higher than the original as-is order because of the time penalty paid for blocking and gating the ordered matrix. Even more indicative of the variability than the high standard deviation, the range is between 3,849 days and 56,700 days. Thus, if the cumulative probability function is calculated for the graph in the above figure, then there is a 0% chance of completing the closures development process on time (i.e., before 1,000 day target). This means that the project will be late 100% percent of the time due to the inherent structure of the system. Quite simply, there is no control in the current process, even if tasks are re-ordered to reduce delays. Also, keep in mind that these numbers are for an interaction value of 12.5%, and are simply being used to determine the level of improvement for subsequent calculations. Regardless, the reordering does nothing to help diagnose or improve the underlying structure. For this purpose, the eigenvalues and eigenmodes must be examined.

Dominant Design Mode

In order to determine the dominant design mode, the first step is identifying the maximum real eigenvalue. The eigenvector associated with this eigenvalue can then be used to identify and diagnose the dominant mode of the system. After determining this controlling eigenvector, which can be viewed as a vector of work percentages, the individual values were weighted by task durations and then used to calculate an overall weighted percent of work. Of this list, the tasks performing the most work are the best candidates for improvement, either by shortening their durations or removing key interactions. For the closures DSM, two scenarios were interesting: first processing just the FC tasks (just product development), and then processing the entire closures DSM (all of product development and vehicle operations).

FC Tasks Only	Percent of Work	Complete Closures DSM	Percent of Work
Prog Strategy - Program Mission And Vision Frozen	10.0%	Virtual Series	3.2%
Prog Strategy - Theme - Design Concepts Defined	5.9%	Product & Process Feasibility	3.1%
Prog Strategy - Finance - ABS Defined	5.9%	Manufacturing Plan	2.9%
Prog Strategy - Applicable Shared Platform Strategy Defined	5.0%	Plant Layout	2.7%
Veh Arch - Overall - Prelim Pkg W/ Target Ranges Defined	3.8%	Develop Discrete Event Simulation Models	2.4%
Prog Strategy - Purchasing - Global Sourcing Strategy Defined	3.7%	Develop Program Specific Bill of Process	2.3%
Prog Strategy - Program Assumptions (PDL) Defined	3.4%	Implement Manufacturing Plan	1.9%
Surf - Ext - Bodyside Surf Defined For FC2	3.0%	Theme - Ext - Multiple Themes Defined	1.9%
Surf - Ext - Initial (Global) Surface Defined For FC0	2.8%	Material Flow Plan	1.8%
Prog Strategy - Mfg Strategy Defined	2.7%	Develop Specific Virtual Assembly Simulations	1.7%
Occ Env - Overall - Prelim Pkg W/ Target Ranges Defined	2.5%	Labor Optimization	1.7%
Closures - Frt Door - Swing Study Results Defined	2.4%	Manufacturing Targets	1.6%
Surf - Glass - Side Glass Plane Defined	2.3%	Preliminary Process Design	1.5%
Closures - Rr Door - Swing Study Results Defined	2.1%	Cost Study	1.3%

Figure 50: Comparison table of top 15 hardest working tasks

As the table shows, there are a few tasks in each DSM that truly dominate the work. For the FC process alone, the top few tasks truly represent the determination of the original targets for the program. The program strategy (defining themes, setting finance targets, determining visions, setting shared platforms, working out a manufacturing plan, etc.), the vehicle architecture and surface, and the swing and glass studies are the key tasks that control the entire convergence of the system. This makes intuitive sense because they are the core factors that determine the shape of the door and how it interacts with the broader vehicle system. Program strategies determine the overall theme for the vehicle – whether it's an SUV or a coupe, what the general design should look like, and how it's going to be made. Vehicle architectures and surfaces are the CAD drawings of surfaces that allow the design group to fit the parts together in space. Finally, the glass and swing studies are dynamic analyses that determine how the door will move in space and whether the window will be able to roll down.

However, the addition of the manufacturing tasks to the complete DSM dramatically changes the behavior of the system. The tasks that originally dominated the FC process are now pushed down by another set of tasks that dominate the combined process. The top few tasks are now concentrated on the translation of the product design into the manufacturing process design, specifically whether it can be produced (virtual series, product & process feasibility, simulation models, cost study), and how to produce it (manufacturing plan, plant layout, bill of process, labor optimization). This set of tasks covers nearly the *entire* manufacturing process, implying that one of the most complicated sets of tasks is being iterated multiple times during the vehicle design process. Naturally, this causes dramatic problems during the development of any vehicle.

Upon further examination of the underlying interactions and structure, it becomes clear that the manufacturing tasks are the real "bottlenecks" of the development process. Whereas a typical program strategy or vehicle architecture task has ~10 precedents and dependents, the manufacturing plan and related tasks have ~40. Essentially, there are many more paths to the manufacturing process development tasks, since every product design change must eventually be vetted against the company's ability to make it. This makes the manufacturing plan and its connections the effective bottleneck for the entire process. Unfortunately, Magma's current system is ill-equipped to deal with the rigors of rapidly producing and evaluating a manufacturing process. Fortunately, there are two methods to relieve the bottleneck.

1. Reduce task durations
2. Move the bottleneck and break interactions

First of all, the durations of the manufacturing development tasks could be improved by creating some improved analytical tools. Conducting a virtual series, simulation, plant layout, or cost study currently takes 8-12 weeks to complete, during which the input data is frozen and outputs are unknown. This represents a large chunk of time in which work is progressing before precedents are known to the remaining tasks. Ideally, Magma should be able to complete an iteration of any one of these tasks in 1 week. This would allow the process to iterate much faster and allow current data to be used in the work, thereby avoided massive rework problems. In concept, this is identical to Microsoft compiling its code nightly. Issues and problems are identified quickly, and work does not progress on broken code. Numerically, reducing the duration of these tasks to 1 week creates a process the following matrices:

DSM	18	1	38	37	36	279	278	311	4	565	564	259
18	40	0	0	0	0	0	0	0	0	0	0	0
1	0.125	30	0	0	0	0	0	0	0	0	0	0
38	0.125	0.125	30	0	0	0	0	0	0	0	0	0
37	0	0.125	0.125	20	0	0	0	0	0	0	0	0
36	0	0	0.125	0.125	15	0	0	0	0	0	0	0
279	0.125	0.125	0	0.125	0.125	20	0	0	0	0	0	0
278	0	0.125	0	0.125	0	0.125	20	0	0	0	0	0
311	0	0.125	0	0	0	0	0.125	15	0	0	0	0
4	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	7.5053	0	0	0
565	0	0	0	0	0	0	0	0	0.125	15	0	0
564	0.125	0	0	0	0	0	0	0	0.125	0.125	30	0
259	0.125	0.125	0	0	0	0	0	0	0.125	0	0.125	20

SD	18	1	38	37	36	279	278	311	4	565	564	259
18	10	0	0	0	0	0	0	0	0	0	0	0
1	0.01875	7.5	0	0	0	0	0	0	0	0	0	0
38	0.01875	0.01875	7.5	0	0	0	0	0	0	0	0	0
37	0	0.01875	0.01875	5	0	0	0	0	0	0	0	0
36	0	0	0.01875	0.01875	3.75	0	0	0	0	0	0	0
279	0.01875	0.01875	0	0.01875	0.01875	5	0	0	0	0	0	0
278	0	0.01875	0	0.01875	0	0.01875	5	0	0	0	0	0
311	0	0.01875	0	0	0	0	0.01875	3.75	0	0	0	0
4	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	0.617+3555i	0	0	0
565	0	0	0	0	0	0	0	0	0.01875	3.75	0	0
564	0.01875	0	0	0	0	0	0	0	0.01875	0.01875	7.5	0
259	0.01875	0.01875	0	0	0	0	0	0	0.01875	0	0.01875	5

Figure 51: Reduced matrices after reducing task durations to 1 week

As the matrices show, the primary change came in task #4. Ultimately, the new matrices produce the following histogram:

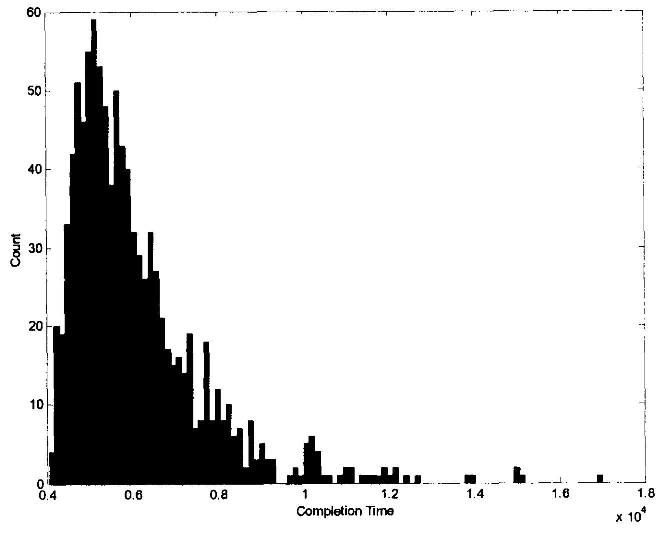


Figure 52: Histogram for reduced DSM with shorter task durations

The new statistics are an average completion time of 6,109 and a standard deviation of 1,613 days. The range of the statistics is smaller, with a minimum of 4,050 and a maximum of 17,485.

The most notable changes after reducing the task durations to 1 week is the dramatically smaller range and standard deviation. Completing tasks faster makes the consequences of iterations be much less drastic, as is intuitively expected. Although improved, this system is still well off its target, even with the reduced assumption of a 12.5% interaction value. Most importantly, none of the duration reductions from improved tools actually change the underlying structure of the system, so the same behavior results.

The second option is to move the bottleneck to the front of the overall process. Essentially, most of the difficulty in Magma's development process occurs when a vehicle is first designed and then fit into a manufacturing plant. This order of execution makes the product design a precedent for the process design, thereby requiring the manufacturing plant to adjust. Obviously, the plant will not be able to exactly make the designed vehicle, and a number of costly iterations ensue as the product and process attempt to converge. Eventually, a measure of convergence is forced at the very end when the launch team is brought on to solve the remaining problems. In the current process, these iterations are really combining two broad sets of tasks into an infinite loop. The process development is not only evaluating the production feasibility of a given design, but also participating in product and process evolution:

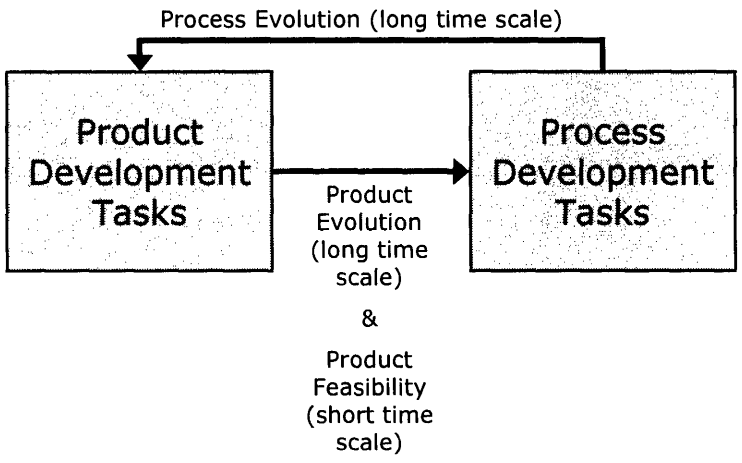

Process Evolution (long time scale)
Product Development Tasks
Process Development Tasks
Product Evolution (long time scale)
&
Product Feasibility (short time scale)

Figure 53: Schematic of primary non-converging loops

As the schematic shows, there are truly two sets of tasks that occur between the product and process development. Unfortunately, these two sets of tasks operate on completely different time scales. Production feasibility and decisions can happen within months, but the joint evolution of product and process capabilities can take years. There is a continuous back and forth between product and process design as designers and engineers balance the desire to make a particular product with their ability to actually make it. Naturally, this product and process combination evolves over time, but it is unwise to combine the evolution into the production decisions. This essentially embeds a longer time scale process in a situation that requires rapid execution, thereby crippling the product development process. Even worse, this evolution is happening separately with each product and process combination in Magma's portfolio, creating a divergent pattern of evolution that reduces commonality and precludes standardization, economies of scale, and the creation of system architectures. The only solution to this predicament is to separate the time scales and activities:

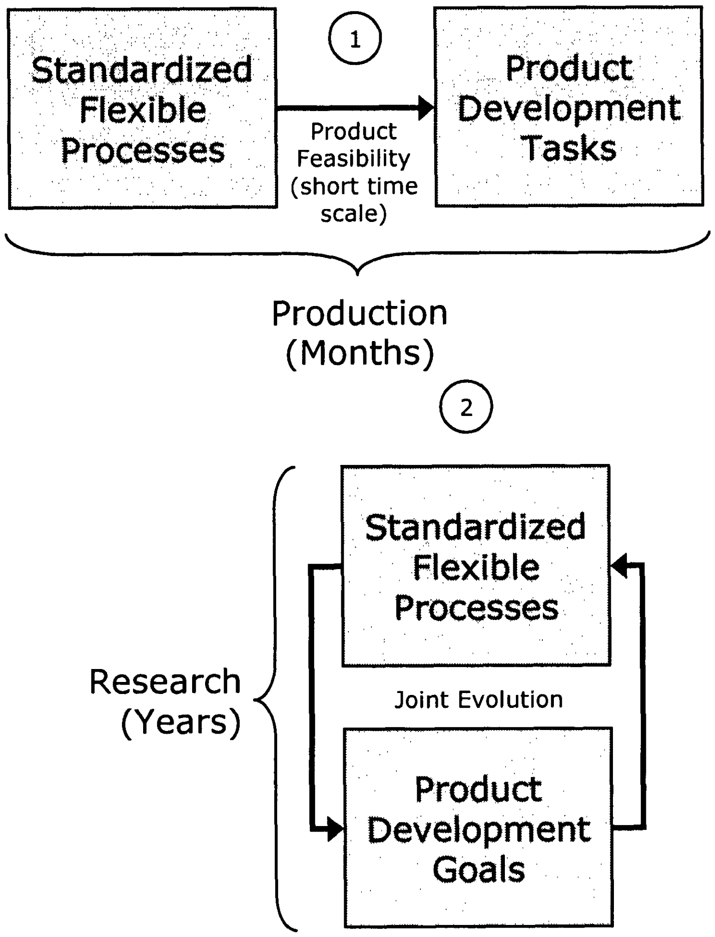

1
Standardized Flexible Processes
Product Feasibility (short time scale)
Product Development Tasks
Production (Months)
2
Research (Years)
Standardized Flexible Processes
Joint Evolution
Product Development Goals

Figure 54: Schematics of separated time scales in development

Fundamentally, the intent of the separation is to split the two time scales present in the current process. For production decisions and rapid execution, is makes sense to choose the manufacturing plant or process first, and then design a vehicle to fit that process. This move breaks the precedent connections from the product design to manufacturing and reverses them. However, this move must be made extremely carefully, since Magma would not want to design its cars to fit a non-standardized and inflexible process that is characteristic of the plant layouts today. Rather, the manufacturing process should be redesigned first and then standardized as a basis for future product designs. This would then allow the second loop to function, evolving the process facilities on a longer time scale with dedicated resources to determining the correct balance of product and process capability across Magma's vehicle portfolio. Presumably, this evolution could also be used to converge Magma's disparate processes into a unified development and production philosophy. Numerically, this change would be the equivalent of eliminating all precedent relationships from manufacturing to PD (or eliminating the manufacturing tasks from the overall closures DSM), which would give the following reduced DSM and histogram:

DSM	1	39	38	37	36	279	278	311	4	270	16
1	40	0	0	0	0	0	0	0	0	0	0
39	0.125	30	0	0	0	0	0	0	0	0	0
38	0.125	0.125	20	0	0	0	0	0	0	0	0
37	0.125	0	0.125	20	0	0	0	0	0	0	0
36	0	0	0.125	0.125	25	0	0	0	0	0	0
279	0.125	0.125	0	0.125	0.125	20	0	0	0	0	0
278	0	0.125	0	0.125	0	0.125	20	0	0	0	0
311	0	0.125	0	0	0	0	0.125	15	0	0	0
4	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	5.2356	0	0
270	0.125	0	0	0	0	0	0	0	0.125	20	0
16	0.125	0.125	0	0	0	0.125	0.125	0.125	0.125	0.125	40

SD	1	39	38	37	36	279	278	311	4	270	16
1	10	0	0	0	0	0	0	0	0	0	0
39	0.01875	7.5	0	0	0	0	0	0	0	0	0
38	0.01875	0.01875	5	0	0	0	0	0	0	0	0
37	0.01875	0	0.01875	5	0	0	0	0	0	0	0
36	0	0	0.01875	0.01875	6.25	0	0	0	0	0	0
279	0.01875	0.01875	0	0.01875	0.01875	5	0	0	0	0	0
278	0	0.01875	0	0.01875	0	0.01875	5	0	0	0	0
311	0	0.01875	0	0	0	0	0.01875	3.75	0	0	0
4	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	0.537+330i	0	0
270	0.01875	0	0	0	0	0	0	0	0.01875	5	0
16	0.01875	0.01875	0	0	0	0.01875	0.01875	0.01875	0.01875	0.01875	10

Figure 55: Reduced DSM for standardized manufacturing

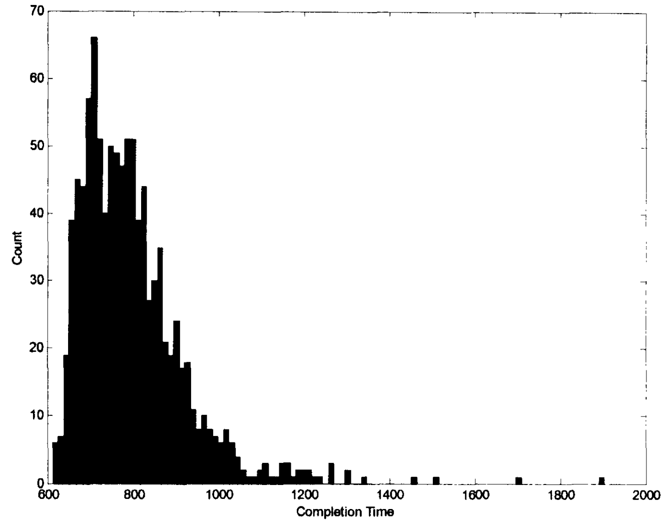


Figure 56: Histogram for standardized manufacturing

The histogram indicates that standardizing the manufacturing architectures produces a dramatically improved performance with an average completion time of 798 days and a standard deviation of 127 days, since most of the manufacturing tasks are removed from the short time-scale production decisions. The range is also greatly improved, with a minimum of 612 and a maximum of 1903 days. Even more importantly, the cumulative percentage analysis reveals that there is a 94% chance of completing the project in less than 1,000 days. Obviously, 94% is clearly better off that the starting position of 0%, implying that there are only a few scenarios in which the project wouldn't be naturally converged by the deadline and would require heavy management interference. If the additional techniques of task duration reduction and eigenvalue tearing were employed on the standardized manufacturing DSM, the subsequent benefits would be even more dramatic.

Finally, implementing the manufacturing changes reduces the core eigenvalue of the system from 0.79 to 0.49, moving the maximum number of iterations from 5 to 2! When scaled up to the more realistic assumption of a 20-25% interaction value, the standardized manufacturing DSM would have an eigenvalue of 0.78-0.98. The good news is that standardizing manufacturing would dramatically improve the process and almost assuredly create a converging system. However, the number of maximum iterations would range from 5 to 50, creating a high probability that the project could still not be completed reliably within 1,000 days. In this situation, another round of standardization, duration reduction, and eigenvalue tearing would be required to further improve the process.

Tearing

After analyzing the dominant mode, it's also useful to take the DSM through the eigenvalue tearing process to figure out which interactions are the best to remove. This analysis could provide alternatives to the major standardization discussed above. The results are in the following table:

Task	**Precendent**	**Resulting Eigenvalue**
Prog Strategy - Theme - Design Concepts Defined	Product & Process Feasibility	0.78275
Product & Process Feasibility	Develop Program Specific Bill of Process	0.77577
Product & Process Feasibility	Manufacturing Targets	0.76852
Product & Process Feasibility	Manufacturing Consensus Review	0.76267
Product & Process Feasibility	Preliminary Process Design	0.75645
Product & Process Feasibility	Process Design	0.75075
Assembly - Process - Flow And Tolerances Defined	Develop Program Specific Bill of Process	0.73223
Manufacturing Targets	Virtual Series	0.72313
Product & Process Feasibility	Mfg - SM - Process - Pdpd Reqmts Defined	0.71651
Manufacturing Plan	Virtual Series	0.71088
Confirm Manufacturing Plan	Virtual Series	0.70848
Implement Manufacturing Plan	Virtual Series	0.70819
Theme - Ext - Single Theme Selection	Product & Process Feasibility	0.70205
Manufacturing Targets	Develop Program Specific Bill of Process	0.69323
Cutlines - Frt Door - Cutlines Defined	Product & Process Feasibility	0.68779
Develop Program Specific Bill of Resource	Develop Program Specific Bill of Process	0.6826
Theme - Ext - Theme Selection For Market Research	Product & Process Feasibility	0.67787
Preliminary Process Design	Develop Program Specific Bill of Process	0.67259
Welding - Weld Patterns, Types, Quality, Class Defined	Develop Program Specific Bill of Process	0.66876

Figure 57: First 20 removed interactions from the eigenvalue tearing process

By design, the best interactions to remove involve the tasks performing the most work in the dominant design mode. This calculation has a slight distinction from the dominant mode above, mainly due to the fact that the eigenvalue tearing process does not weight the eigenvector by the task duration times. Rather, the pure eigenvector is used to determine the most active tasks and process the appropriate set of interactions for reducing the eigenvalue. Regardless, it is interesting to note that relatively few interactions are required to make a significant dent in the eigenvalue of the system. The first ten tears reduce the eigenvalue from 0.79 to 0.71, which represents a reduction in iterations from 4.76 to 3.45. After the tears are calculated, it's useful to examine each dominant mode and determine the hardest working tasks as the interactions are removed. Tracking the top 10 tasks over the first 20 tears gives the following chart:

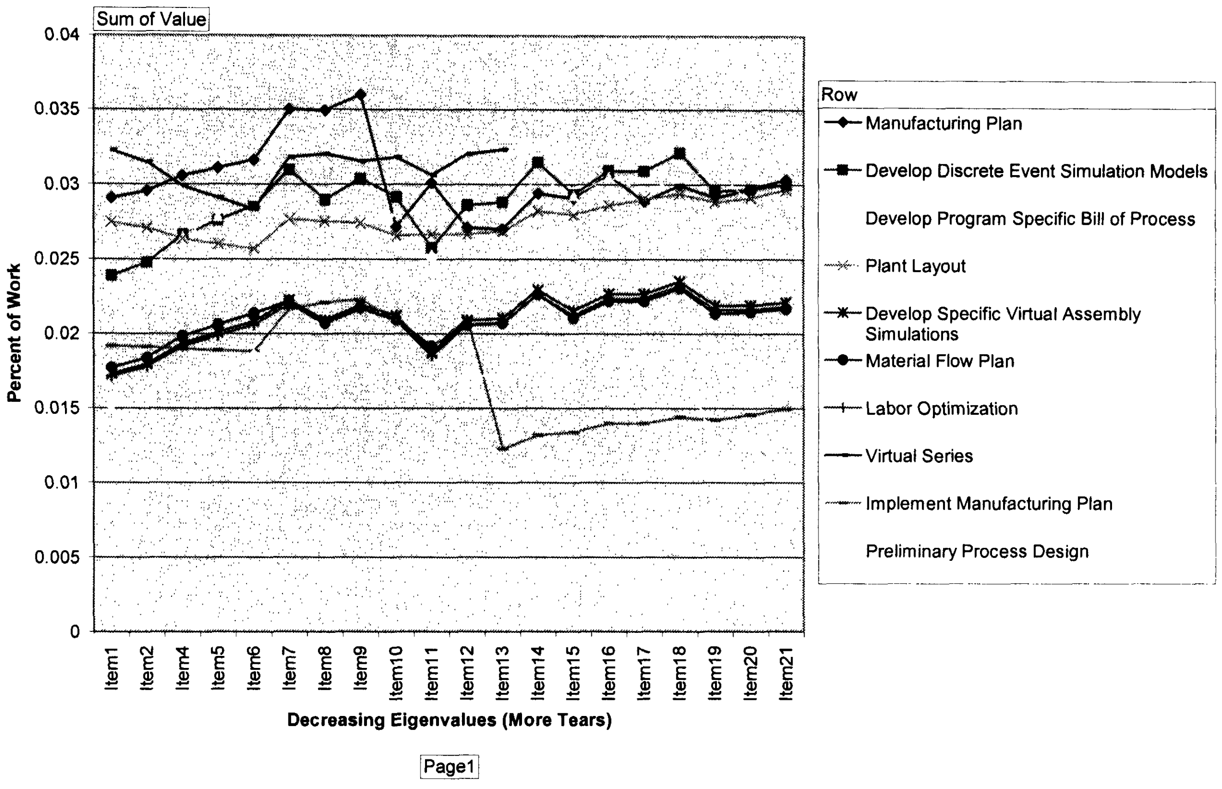


Figure 58: Chart of hardest working tasks as 20 eigenvalue tears are removed

As the chart shows, the hardest working tasks are fairly robust to the removal of interactions. This is largely due to the fact that most of the tasks up in the top 10 have a vast number of connections into the overall system. As a result, the simple removal of one or two interactions will have little effect on the amount of work being fed to them as bottlenecks in the system. The only exception is the virtual series, which drops out of the top 10 as its key interaction is removed in the 14th tear. Examination of this data set confirms the interpretation of the initial dominant mode analysis. Since the major manufacturing tasks continue to be vital to the system performance, removing a small number of interactions will not render the DSM to be a tractable problem. In fact, nearly the entire integrity of the central 391 task parallel block is being retained. As a further example, here is the reduced DSM with the first 50 tears removed and a resulting eigenvalue of 0.62:

DSM	18	1	38	37	36	279	278	311	4	555	564	259
18	40	0	0	0	0	0	0	0	0	0	0	0
1	0.125	30	0	0	0	0	0	0	0	0	0	0
38	0.125	0.125	30	0	0	0	0	0	0	0	0	0
37	0.125	0.125	0.125	20	0	0	0	0	0	0	0	0
36	0	0	0.125	0.125	25	0	0	0	0	0	0	0
279	0.125	0.125	0	0.125	0.125	20	0	0	0	0	0	0
278	0	0.125	0	0.125	0	0.125	20	0	0	0	0	0
311	0	0.125	0	0	0	0	0.125	15	0	0	0	0
4	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	5.5425	0	0	0
555	0	0	0	0	0	0	0	0	0.125	20	0	0
564	0.125	0	0	0	0	0	0	0	0.125	0.125	30	0
259	0.125	0.125	0.125	0.125	0	0	0	0	0.125	0.125	0.125	20

SD	18	1	38	37	36	279	278	311	4	555	564	259
18	10	0	0	0	0	0	0	0	0	0	0	0
1	0.01875	7.5	0	0	0	0	0	0	0	0	0	0
38	0.01875	0.01875	7.5	0	0	0	0	0	0	0	0	0
37	0.01875	0.01875	0.01875	5	0	0	0	0	0	0	0	0
36	0	0	0.01875	0.01875	6.25	0	0	0	0	0	0	0
279	0.01875	0.01875	0	0.01875	0.01875	5	0	0	0	0	0	0
278	0	0.01875	0	0.01875	0	0.01875	5	0	0	0	0	0
311	0	0.01875	0	0	0	0	0.01875	3.75	0	0	0	0
4	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	0.01875	0.90+1085i	0	0	0
555	0	0	0	0	0	0	0	0	0.01875	5	0	0
564	0.01875	0	0	0	0	0	0	0	0.01875	0.01875	7.5	0
259	0.01875	0.01875	0.01875	0.01875	0	0	0	0	0.01875	0.01875	0.01875	5

Figure 59: Reduced DSM with 50 tears removed

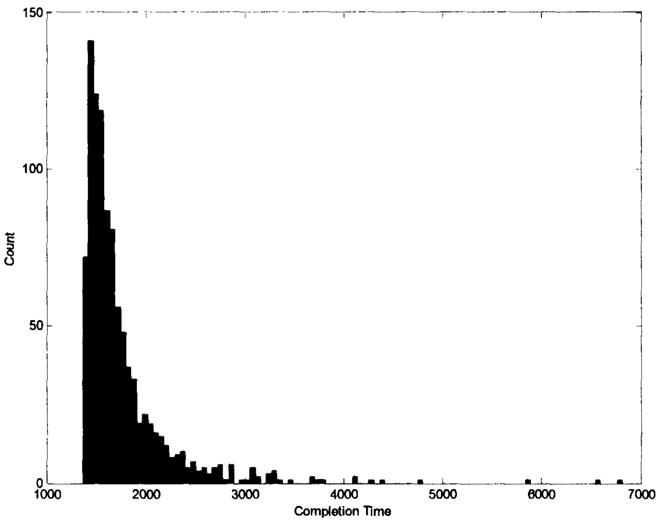


Figure 60: Histogram with 50 tears removed

As the new data shows, the new results are an average is 1761 days, a standard deviation of 493 days. The new range is from a minimum of 1365 to a maximum of 6814 days. Consequently, there is still a cumulative probability of 0% of finishing before 1,000 days. Removing 50 interactions has clearly had an effect on the system, but not nearly enough to cross the 1,000 days threshold or reduce the standard deviation to a reasonable level. Without a doubt, removing more interactions will improve the DSM. Tearing out these extra interactions also allows a significant amount of reordering to take place, which reduces the coupling and allows smaller parallel iteration blocks to be formed.

However, a simple examination of the torn interactions reveals that they would be nearly impossible to eliminate. For example, connections between program strategy and product & process feasibility are fundamental to the current development process, and cannot be removed without more basic changes to the structure. In fact, after examining the top connections, it seems clear that the only way to eliminate this set is to apply the same standardization of manufacturing processes described above. That would effectively remove the tasks altogether, which obviously removes all of the associated interactions.

Summary

The statistics after each major change to the DSM are shown below:

	Average	SD	Min	Max	Eigenvalue	Percent Chance <1000 days
Re-ordered and Reduced	7345	4815	3849	56700	0.79	0%
Faster Key Task Durations	6109	1613	4050	17485	0.79	0%
Standard Manufacturing	798	127	612	1903	0.49	94%
50 Interaction Tears	1761	493	1365	6814	0.62	0%

Figure 61: Summary table for DSM modifications

Overall, the simulation results show that reducing task durations, removing significant interactions, and eliminating tasks all have beneficial effects beyond a simple re-ordering of the matrix. However, the only improvement that is large enough to be meaningful to Magma is the standardized manufacturing. The other improvements, although mathematically significant, would only serve to reduce the number of hidden errors in a non-converged design that is passed on once the deadline of 1,000 days is reached. This clearly shows that standardizing manufacturing is the best alternative to improve the development process performance, and it should be treated as the fundamental improvement, whereas the other changes can be secondary.

Resource Model

Regardless of the eventual modifications to the closures development process, the resource allocation process can definitely be improved. Unfortunately, resource data for each task was not collected as part of the work at Magma, so each task was assigned a resource requirement of 1. Running the standardized manufacturing DSM through the resource variation routine gives the following plot, covering starting resource levels of 50% through 250%:

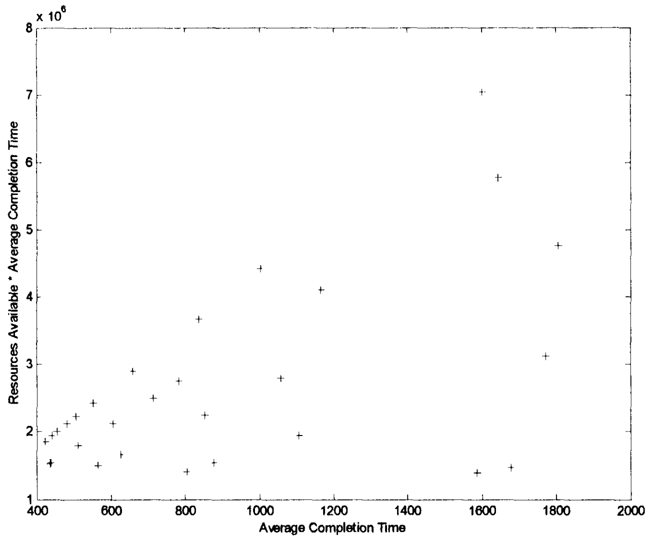


Figure 62: Resource model results for standardized manufacturing DSM

As the figure shows, there is a noticeable lack of resource relaxation curves. Rather, the plot is primarily showing the effects of shifting utilization factors, and forming the horizontal line at the bottom of the frontier. Of course, the effect of going over a factor 2 is also shown in the highest line being a simple increase in cost with no consequent benefit of reduced completion time. The resource relaxation curves are missing because the DSM's are being processed in their reduced form for computational reasons (i.e., the effects of iteration are imbedded in the lognormal distribution assigned to one task) and because of the required resource profile in the reduced DSM:

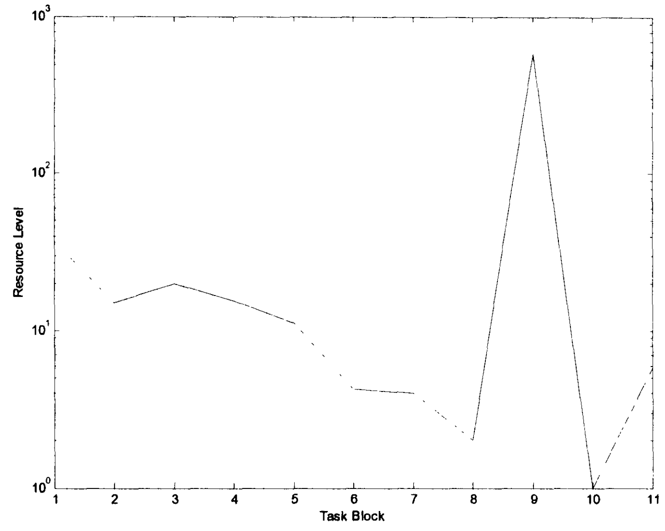


Figure 63: Resource profile for reduced DSM

Overall, the reduced DSM is grouping a lot of independent parallel tasks in the beginning of the order and also satisfying a large parallel block in the middle, and thereby requiring a large number of resources working on the project. As long as these resources are present and allocated to the project, there will be no resource constraints for any other tasks, thus explaining the lack of the resource relaxation curves. There are simply no resource needs in these reduced DSM's. In contrast to the two high resource level requirements, the other tasks either represent smaller groups of tasks that require smaller resources or large coupled blocks that work the tasks in the simulated sequential fashion. Thus, the other coupled blocks only require a key set of resources (i.e., a core team) that works together on the coupled tasks.

In spite of the limited behavior of the single project, simulating the resource requirements for multiple programs does generate the anticipated cost and resource behavior. First, the optimum delay between two projects had to be determined:

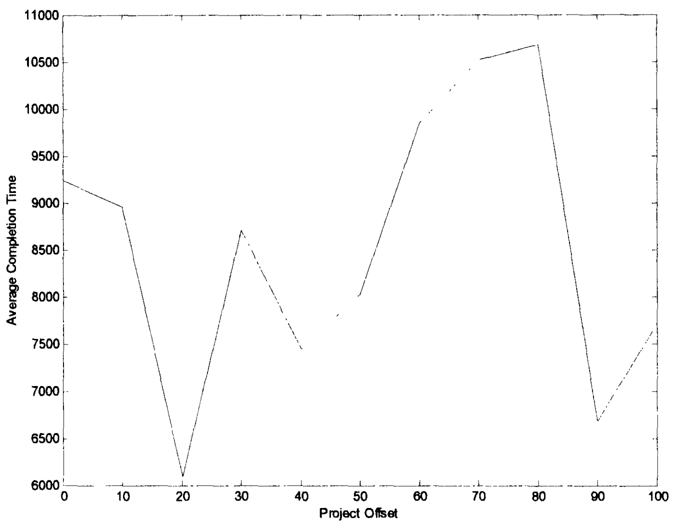


Figure 64: Plot of completion time versus delay offset for two projects

Clearly, the delay of 20 days between projects provides the best combination of the projects for destructive interference under a resource constraint. Using this delay in the resource variation simulation then generates the following plot:

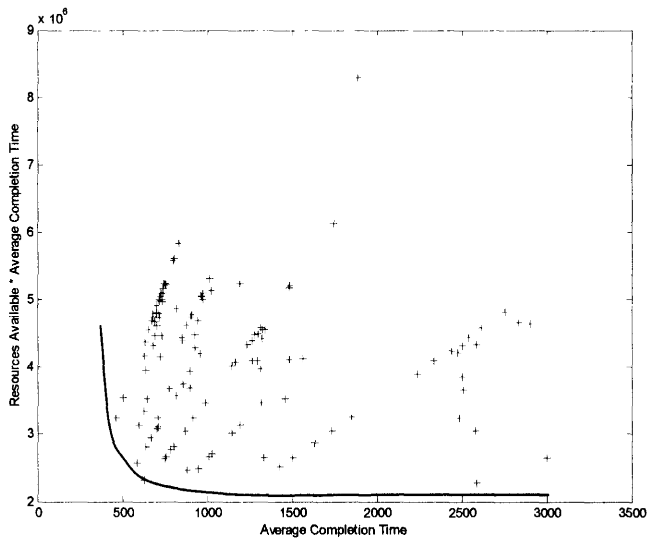


Figure 65: Multiple project resource frontier

Overall, this resource use frontier resembles the one developed for the example DSM earlier. Again, the frontier can be used with a budget constraint to determine the optimum point of operation for Magma. The frontier does have a relatively sharp curve in this case, which suggests that the range of optimal points will be close together. However, each one of the optimal points will be above the set of resource constrained points that make up the horizontal line at the bottom of the frontier, assuming that a value of time greater than zero is used.

Presuming that Magma has a value of time greater than zero, staffing at the optimum resource level would allow them to adequately handle all of the significant iteration that occurs in the middle of the larger parallel blocks. However, Magma's general staffing levels would have to change significantly in practice, since Magma isn't accustomed to (1) retaining people on projects for longer than 2 years, (2) having uneven loading of projects, and (3) being overstaffed. The uneven loading could easily be dealt with by staffing employees on two staggered programs at once as modeled by the offset delay, thus allowing them to build experience on both while using the staggered development schedule to balance their workload. Unfortunately, resolving the 2-year rotational issue is a much larger organizational problem. Magma has been accustomed to promoting and rotating personnel on a 2-year schedule for ~50 years, and is only beginning to value technical maturity on projects. Finally, an overstaffed situation would be incredibly unlikely in a financially driven culture.

Error Rates and Early System Involvement

Finally, addressing the error rate implicit in any development activity could go a long way to improving Magma's position. Following the error model described earlier, it's useful to assume that 5% of the 12.5% interaction value is actually due to errors in the development process. If this error rate were to be eliminated, the interaction value would fall to 7.5%, thereby creating an eigenvalue of 0.47. This new eigenvalue means that only 1.89 iterations would be required to reach completion instead of 4.76. This is a savings of over 60% in completion time, achieved by staffing a project early and focusing on discovering errors and system issues well before launch. However, as noted before, the level of improvement is highly dependent on the starting position of the system.

Magma's Sensitivity to Expected Interaction Values

As a summary, it is useful to remind oneself that all of the above calculations were performed using a 12.5% interaction value to generate a starting eigenvalue of 0.79. This was done to ensure that errors would not be present in the mathematics owing to eigenvalues greater than 1.

However, the lessons learned above will be direct analogues to Magma's actual process, no matter what interaction value is expected. In reality, the interaction value for Magma's closures DSM is probably closer to 20-25% (probabilities of 32-40% with an impact of 62.5%), which seems to be more in line with the types of tasks being analyzed. Unfortunately, these numbers imply that Magma actually has a naturally non-converging process with an eigenvalue of 1.3-1.6. From this starting position, reducing task durations would have no impact whatsoever. If a process cannot converge, it does not matter how fast iterations are completed because they will last *forever*. The only way to exit from a non-converging loop is to willingly forego interactions and loops, or by finishing work with incomplete information. In practice, management is required to intervene in order to force the process to converge by a given deadline. However, this forcing is likely to produce hidden errors in the product as decisions are made with incomplete or non-converged information. As evidenced by the large volume of launch issues on most Magma programs, this is most likely what's happening in the current process. Therefore, reducing task times is simply not a solution.

Furthermore, success is still difficult even if Magma eliminates all errors in their current process. If Magma pursues the systems outlook and early error discovery track, the interaction value could fall to a range of 15-20%. However, this makes the unlikely assumption that execution is perfect, with no errors whatsoever. At this perfect level of execution, the eigenvalue would then be 0.95-1.26, implying that there is a chance for the process to converge. However, an eigenvalue of 0.95 suggests that 20 iterations would be required to reach convergence. Operationally, this is entirely unfeasible because the time required to develop a converged design would be orders of magnitude greater than breaking the non-converging process today. In addition, no one wants to do the same job 20 times for one vehicle program. Even combined with the task duration improvements above, the process would still remain intractable. In fact, very little change would be evident from today's process, since interactions would still have to be skipped to meet the imposed development schedule. Launch problems would still happen, and the unproductive loops would continue.

However, if Magma undertakes the standardization recommendation and moves manufacturing activities to be the unidirectional inputs to the product development, the entire structure of the system changes. In this scenario, the eigenvalue becomes 0.78-0.98 even before the error reduction and potential eigenvalue tearing. When combined with the error reduction of 5% to reduce the interaction value to 15-20%, the result becomes 0.59-0.78. At this range, the closures development process is becoming a tractable problem. The maximum number of iterations would be 2.5 to 5, which is vastly improved over the initial non-convergence. At this level of performance, additional task duration reductions become less important since the level of repetition is lower. However, the ability to process virtual series, simulations, and feasibility studies faster would allow the system to complete its iterations that much faster and rapidly reach convergence.

Combining all of the improvements so far for the higher estimate of Magma's interaction value, the following DSM and histogram representing a solution involving standardized manufacturing, error reduction, and improved analytical tools are below (with an expected interaction value of 20% - composed of 32% probabilities and 62.5% impacts):

DSM	1	39	38	37	36	279	278	311	4	270	16
1	40	0	0	0	0	0	0	0	0	0	0
39	0.2	30	0	0	0	0	0	0	0	0	0
38	0.2	0.2	20	0	0	0	0	0	0	0	0
37	0.2	0	0.2	20	0	0	0	0	0	0	0
36	0	0	0.2	0.2	25	0	0	0	0	0	0
279	0.2	0.2	0	0.2	0.2	20	0	0	0	0	0
278	0	0.2	0	0.2	0	0.2	20	0	0	0	0
311	0	0.2	0	0	0	0	0.2	15	0	0	0
4	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	7.2048	0	0
270	0.2	0	0	0	0	0	0	0	0.2	20	0
16	0.2	0.2	0	0	0	0.2	0.2	0.2	0.2	0.2	20

SD	1	39	38	37	36	279	278	311	4	270	16
1	10	0	0	0	0	0	0	0	0	0	0
39	0.03	7.5	0	0	0	0	0	0	0	0	0
38	0.03	0.03	5	0	0	0	0	0	0	0	0
37	0.03	0	0.03	5	0	0	0	0	0	0	0
36	0	0	0.03	0.03	6.25	0	0	0	0	0	0
279	0.03	0.03	0	0.03	0.03	5	0	0	0	0	0
278	0	0.03	0	0.03	0	0.03	5	0	0	0	0
311	0	0.03	0	0	0	0	0.03	3.75	0	0	0
4	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.855+595i	0	0
270	0.03	0	0	0	0	0	0	0	0.03	5	0
16	0.03	0.03	0	0	0	0.03	0.03	0.03	0.03	0.03	5

Figure 66: Final reduced DSM for Magma's closures development process

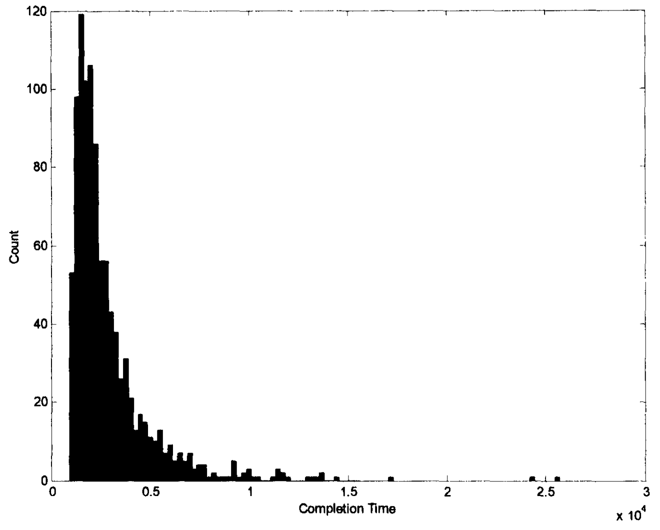


Figure 67: Final histogram for standardized, error-reduced, analytical tools process

As the above chart shows, using the standardization, error reduction, and improved tools can reduce the closures development time, even with the increased probabilities in the DSM. The above distribution has an average of 2,932 days and a standard deviation of 2,267 days, with a minimum of 921 days and a maximum of 25,685 days. Overall, this distribution has a 1% chance of completing the process in less than 1,000 days. Unfortunately, it seems that Magma may still have a process that cannot meet their goals without management intervention, even with all of the improvements discussed above. However, it is important to note that the process would not converge at all unless the improvements were made. Undoubtedly, the above distribution represents progress. Of course, the distribution also illustrates the importance of data accuracy in these sensitive calculations. An assumption of 12.5% interaction values generates an average completion time of 798 days, while an assumption of 20% generates 2,932! The relationship is clearly non-linear, and that may the most important lesson of the entire analysis.

In summary, it is vital to point out that these modifications to the structure of the closures development process create a much greater chance of completing the process on time. The changes also help to reduce the level of management interference that is required to force convergence and actually roll a product off the line on time. However, there will nearly always be a finite chance of that the project will not be completed on time, or will have hidden errors, even after all of the improvements. In other words, it is possible that a closures development project with an average time of 798 days will not be able to converge in the allotted time, management will have to get involved to force the situation, and launch issues will result. Most importantly, this failure to converge is no one's fault; rather, it is the consequence of the system's structure. Quite similarly, a project that converges in 600 days isn't necessarily the result of anyone's skill in project management, either – probability plays a huge role in the outcome. In fact, all of the early completions in Magma's history must be due to breaking the structure, skipping iterations, and neglecting interactions, because the analysis suggests that the development would never finish in time if it were allowed to complete naturally. The project manager, or any employee for that matter, is bound to the execution of a development structure that will inherently vary in its performance, thereby making accurate prediction and execution of the closures development process nearly impossible. For this reason, managers should be focusing on iterations and interactions that drive convergence and *performance will be a logical result.*

Organizational Implications from the DSM Analysis

After a DSM has been processed and reordered, there are several organizational implications that arise from the new structure. Most importantly, the task reorder completely changes the path that a development process takes to completion. Obviously, this change in order can have severe effects on the jobs of employees and on the power culture determined by the flow and control of information in any intellectual process. However, the development of the DSM produces several insights that must be true for a successful implementation:

1. Follow the new order
2. Use analytical tools to speed iterations
3. Do not begin work on a subsequent block before the preceding block is complete
4. Assign people to manage parallel blocks: interactions, convergence, and completion
5. Manage information transfer between programs and through time

The first recommendation is straightforward. If the new order of tasks isn't followed, then the feedback loops will still have the original delays in them. Simply realizing that the major iterative loops have been identified and instructing the organization to be more disciplined in their execution will have absolutely no effect on the process whatsoever.

The second recommendation is also relatively straightforward, and its implication is primarily based on the use of analytical tools as facilitators in the development process. Tools such as CAE, virtual series, and cost studies are designed to help identify and fix system design issues. In reality, Magma's current tools are acting as significant delays in the development process while data is frozen, assumptions are made, calculations are performed, and answers are finally returned to the original process. In the meantime, work is either on hold or progressing and creating rework when the results of the studies finally come out. High priority needs to be given to making these tools run faster, because they are bottlenecks for the development process. In reality, a faster tool may be nearly impossible because of the huge number of variables that need to be covered for an accurate analysis. For this reason, some simplifying architectures need to be developed in line with manufacturing standardization and other system changes. The ultimate goal should be studies and tools that can run to completion in a matter of days, not weeks. This will allow the organization to focus on completing iterations and developing a converged design, not on the tool itself. Keep in mind that none of these tools change the structure of the development process, so they can only hold up the iterations while everyone waits for their results.

The third and fourth implications are tied together in a more complicated relationship than the first two. From the example DSM, the blocking of tasks established which tasks could be executed in parallel:

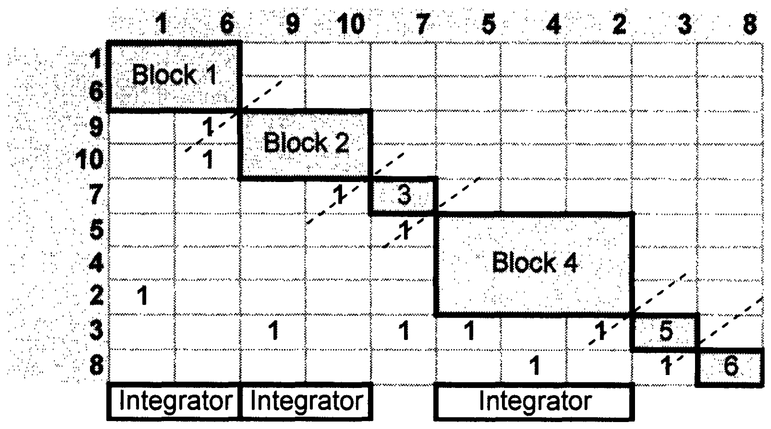


Figure 68: Example DSM with parallel blocks, dashed gates, and integrator assignments

In the above DSM, six parallel blocks have been identified during the reordering process. Each of these blocks was created to encompass related tasks that could be, or had to be, run in parallel while also only delivering information downstream to the remaining blocks. In this manner, there would be no feedback between major blocks in the system. These blocks also provide natural gates for the overall development process to have sign-offs and reviews in the appropriate places. However, if this system is violated, much of the benefit of the DSM analysis can be destroyed. If work begins on block #2 before block #1 is complete, some work will be done before the precedents are complete. Consequently, when block #1 does complete, the new information will need to be incorporated into block #2 and some rework will need to be done. Obviously, this means the initial work done on block #2 was a waste of time and actually prevented those resources from working on something else constructive. In other words, the block was not working ahead, and attempting to do so is foolish. By extension of this rationale, it never makes sense to overlap work on successive blocks. However, this rigidity does prevent some of the tasks that would normally work ahead without consequence from doing so.

Unfortunately, the current FC process often falls victim to violations of the process gates. When the original feasibility checkpoints were created, they were placed at the end of large parallel blocks with the intent to gate the development process and prevent rework. In practice, these gates are often violated when work isn't complete by a given checkpoint. The project managers decide to move through the checkpoint and begin work on the next block while the current one isn't finished. Their intent is to get ahead on the remaining work while developing a road map to finish the incomplete block. However, it is clear that they are only creating rework for themselves. As a result, the enforcement of the FC process, or any DSM-related gate, is vital to receive the performance gains. Much of this problem could be solved by increased training on the FC process. Much of the original FC work was done and forced into the current MPDS process without adequately explaining the reasons for the changes. So, as far as the engineering staff is concerned, this is simply another set of deadlines to meet. The primary failing of the lack of training is the understanding that the development process is not strictly linear and cannot be forced into a Gantt chart.

The fourth implication is primarily an outgrowth of the complicated nature of product and process development. When large groups of people are required to work in parallel to develop an idea, it would be ideal to have them all around a large table in a conference room until they come up with a solution. Being in the same location greatly enhances communication, and often explains why smaller programs in Magma can execute a development process so well. Practically, the matter is much more complicated with a full program. The number of people involved on any full-scale program is far too large to throw into one room. Consequently, it makes sense to have systems integrators in charge of each parallel block as the development process moves along. The integrator's role would largely consist of facilitating communication and interaction amongst parties in the parallel blocks. For this reason, they would need to be given the responsibility for convergence of the parallel block, and the authority to enforce decisions within the political environment. System integrators cannot simply advisors to the process, or have the responsibility with no authority. Their responsibility is to control interactions, make trade-offs, and generally develop a converged design. For this reason, they must have power over the process and those people making engineering decisions. By extension, any integrator must also not be beholden to their original functional departments. Individual budgets, staffs, and hierarchies will likely be required to force a separation between functional creation and systems integration.

Unfortunately, the current system integrators are largely mired in firefighting of launch issues and broader conflict resolution on a variety of programs. Very little is being done in the way of systems integration on new programs or parallel blocks. In addition, the closures system integrators continue to be paid by their original functional departments and fall under their old supervisors. This robs them of much of the power that could be given to an integration role, and leaves them with all of the responsibility and none of the authority. As a consequence, the system integrators have been structured as "first among equals," which gives advisory power but little sway over final decisions. In the end, many of the current system decisions have been overturned by their old functional managers, resulting in more effort to reach the old answer.

The fifth organizational implication is derived from extending the DSM analysis beyond its current scope of a single program's closures development. In reality, a closures DSM is simply part of a larger development process, which is also part of a series of developments for a given vehicle, which are part of a set of vehicles that Magma markets to consumers. Graphically, this four dimensional problem is represented as follows:

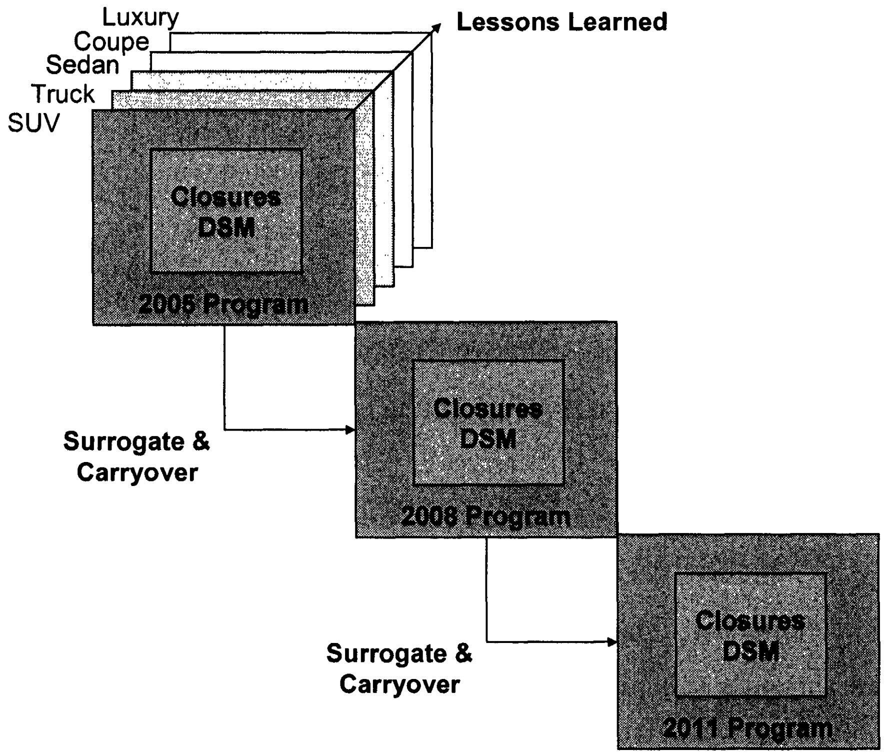


Figure 69: Basic diagram of vehicle development through DSM's, including information flows

As the diagram shows, there are two major flows of information within Magma's broader development programs. First of all, information in the form of company wide architecture and standards are passed between vehicle programs. Often, this information can include generic lessons learned that affect product or process design. Secondly, system and architecture changes are incorporated into each vehicle, which is then used as a surrogate design for the next major development cycle of the vehicle program. Surrogates are actually chosen for every vehicle program, even if the vehicle is theoretically designed from the ground up.

In practice, the transfer of information between programs is often quite low. Programs are developed first as product designs, each based off of their own surrogate vehicle, and then a manufacturing plant is customized for production. As a result, each product and process combination is unique to the vehicle program, and there is a noticeable lack of common architectures and systems across Magma platforms. Obviously, the lack of common architectures precludes much of the information sharing that could occur across platforms. Without a common manufacturing system and process, much less common vehicle platforms, the information sharing must be limited to generic financial and program management items. Furthermore, unique process and product combinations totally prevent any load-leveling from producing the two vehicles in the same plant. All of this leads to a divergent evolution of product and processes, breeding increasing complexity over time.

In contrast, the use of surrogate vehicles is much more common. The intent of a surrogate is to provide a basic engineering platform to design the next major iteration of a program. The surrogates are often chosen to be similar in function, size, and market for the next vehicle in order to maximize the amount of useful carryover designs. Unfortunately, Magma's inability to completely solve systems issues on any given program until launch generates last minute fixes or partially complete solutions. As a result, many of the larger systems issues, such as persistent water leaks, wind noise, and closing efforts remain embedded in the base design. This design is then used as the surrogate vehicle for the next program development, setting the stage for repeat problems. This issue is often complicated by a lack of iteration on the part of the designers, as they will simply pass along the surrogate design in many cases. In this manner, Magma has managed to face the same systemic problems for decades in its vehicles. Quite simply, true solutions are never created, and the problems perpetuate from program to program via vehicle surrogates.

This same continuity from surrogates can also be used to generate a positive evolution for vehicle programs. If Magma is able to solve a system issue in one generation of the program, that change will inevitably be incorporated to the vehicle and passed as part of the surrogate. However, it is vital to point out that evolution is part of the problem that got Magma into its current situation. Evolution and the creation of system solutions to incorporate into surrogate vehicles should really be done from a common product and process architecture; otherwise, the separate evolution of vehicle programs will increase the diversity of solutions. Essentially, the performance of any new vehicle program is path dependent on the chain of surrogate vehicles. Magma should use this dependency to pursue a converging set of solutions, not divergent.

DSM Recommendations and Financial Implications

After processing the closures DSM, the following recommendations are clear from the analysis:

1. Separate time scales by standardizing flexible manufacturing processes for production decisions, and design cars to those processes
2. Address hidden errors with early systems thinking and experience
3. Staff programs according to the new resource model, and retain experienced personnel
4. Reduce task durations where appropriate with analytical tools
5. Address organizational implications of the DSM framework

It is vital to understand that recommendations #2-5 are useless without #1. Unless Magma addresses the fundamental non-convergence of its current process, changes made at the fringes of the problem will never be felt or seen in results. However, it is also important to note that performance is likely to get worse before it gets better. Developing standards, new tools, and changing the staffing process will be expensive, and new processes take time to learn. However, reaching the new equilibrium will be well worth the effort.

The best estimate of savings can be gained by actually applying the budget constraint to the resource curve developed in Figure 65 for two simultaneous projects and an assumption of 12.5% interaction value. The initial estimate of the value of Magma's time can be gleaned from a simple financial analysis. With annual revenues around \$200B, approximately 40 active programs, a net margin of 5%, and closures making up ~25% of the total development work, Magma's value of time could be $200B / 365 days / 40 programs * 5% * 25% = $170K per day for the use of resources in closures development. Also, the calculated curves need to be calibrated to a best estimate of current costs. Considering that two programs currently use ~60 people for a target of 1,000 days at a salary of $100,000 per 250 days, scaling up to the starting point of the curve (factor of 1, resource constrained) would place the cost at 60 * 2550 * $100k / 250 = $61M if they were to finish the program to convergence. Combining these two rough analyses generates the following plot:

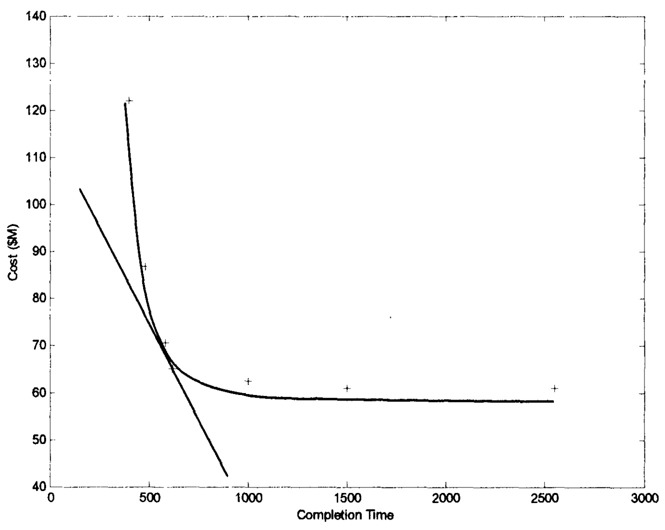


Figure 70: Budget constraint analysis for Magma

As the figure shows, the optimal point for Magma is paying ~$70M for a process that takes ~580 days to converge two programs. This represents an extra expenditure of $9M that saves time worth (2550-580) * $170k = $335M. In practice, Magma would only the value of time saved from its current process, since quantifying the hidden errors and time to market is more difficult. Thus, the value would be (1000-580) * $170k = $71.4M. Obviously, the return on the investment for two programs is certainly worth the effort. If this simple calculation were scaled up to the total 40 programs and to the rest of the development process (not just the 25% for closures), the base value would be worth $71.4M * 20 * 4 = $5.7B. Although this number is clearly inaccurate and not discounted for the time value of money, it provides an order of magnitude estimate for the improvements from the DSM-based recommendations. Undoubtedly, the benefit is even higher due to the avoided hidden errors and potential recalls after a product is launched with design that has been forced to converge.

Unfortunately, the challenges inherent in completely revamping the product and process development process make a goal of billions of dollars of annual savings quite difficult to reach. The level of investment to standardize manufacturing, reduce task durations through system tools, and set policies is also in the billions, and quantification of those items will not be attempted here. Rather, this calculation of potential savings could be viewed as a continuing penalty that Magma pays for having a development process that is inherently flawed.

Implementation Challenges

In spite of the clear recommendations and direction established for Magma, much of the past work has been faced with substantial organizational barriers to implementation. In discussions with previous analysts, three primary blocks were described:

1. Lack of authority or official change process
2. No incorporation into a formal process
3. Lack of continuity

First of all, many of the previous change initiatives have fallen victim to a simple lack of authority in Magma's rigid hierarchy. If a director or VP discovers that there are errors in the process, the scope of the problem is often out of his immediate control. By itself, this is not unusual, since many change programs have been built on the back of consensus and convincing. However, the numbers of levels that must be covered to change a process often result in solutions that must be approved by the CEO and COO in order to be enacted. This is simply inefficient, and is a situation in which Magma's rigid hierarchy truly resists change. This problem is all the more surprising considering that Magma pays strict attention to the change process for engineering decisions, but gives little credit to an official change process for development itself.

Secondly, many past studies failed because of the lack of a formal process to support the execution of the new method. Typically, this happens on any given program when someone develops a tool or checklist that is used to solve a local problem. However, this tool is not incorporated into anything formal and is lost on the next program, or not transferred amongst programs. Often this can be a Catch-22, because no one wants to incorporate a new process until it's proven to work, but proving it requires running two processes at once. Obviously, this doubles the workload of those involved, and typically ensures the death of the new process. There is some hope for Magma with the rise of GPDS (the Global Product Development System), which should be an excellent window to incorporate new lessons into the processes that will be replacing MPDS. However, Magma must assure itself that GPDS is not being bent to fit within MPDS.

Finally, the most common reason cited for failure of change initiatives was a lack of continuity. Quite simply, the people who championed the change are promoted or leave the company. When combined with the lack of incorporation into a formal process, their departure immediately deflates the change process and returns the system to its old ways. The most obvious culprit in this problem is Magma's 2-year promotion cycle that always removes employees from their previous projects in order to promote them. In an environment that has a 4-year development cycle at minimum, it is completely counter-productive to promote and rotate project leaders on a 2-year basis.

Final Context for Product Development: Real Options

As anyone in a long product development process will point out, it isn't always wise to commit to a design too early in the process. Many things can happen in the intervening 4-5 years between concept and launch, including massive market changes in tastes, prices, and competition. As a result, there is a real trade-off to be made between the benefits of early specification and the benefits to waiting. This trade-off can be analyzed using real options to determine the optimal point to begin project convergence.

As discussed in *Ford and Sobek,* the value of project follows three central hypotheses:

1. As the delay to convergence initiation time increases, project values increase as uncertainty decreases

2. As convergence initiation time approaches the deadline, project values decrease development of multiple options becomes costly and less useful in dealing with decreasing uncertainty
3. As unit value of quality increases relative to development cost, the convergence initiation time moves later

From these hypotheses, it's simple to realize that there is an optimal time to begin convergence of a project in order to balance the trade-offs inherent in a real-options analysis. Ideally, the project plan would be calculated by establishing the deadline, backing up the convergence time, and then allowing for convergence delay to process a set of multiple options. This would establish a finite timeline for a project as follows:

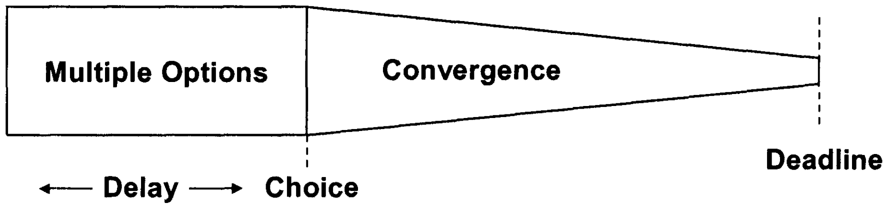


Figure 71: Idealized project management representation of real options (*Ford and Sobek*)

In this view of project management, DSM tools would be used to dramatically reduce the time to convergence, thereby shortening the second segment of the timeline and allowing projects to begin closer to their deadline. Presumably, this would have beneficial effects not only for the fewer resources required to converge, but also in the reduced uncertainty in starting a project closer to the deadline. In addition, the shorter development cycles would also have substantial market benefits in terms of time-to-market and product evolution. All in all, the business case for using DSM results to improve the development process is sound. Valuation of the benefits could be found using the Black-Scholes formula for option valuation, but that is beyond the scope of this thesis.

IV. Organizational Analysis

Any set of recommendations from a technical system analysis would be remiss without an accompanying assessment of the organization. Organizations are simply sets of people working toward a common goal, and they should always be designed to fit the associated strategies for the marketplace. Consequently, it is simply foolish to recommend sweeping changes in processes and functions without first recognizing the broader implications of these changes to the overall organization and the chosen strategy. In light of this concern, the structural, political, and cultural aspects of Magma's organization must be examined before a path forward can be determined. Naturally, this analysis is based on the work described above, but will be applied across the organization as appropriate. In addition, most of the insights are generic to large bureaucratic, functional organizations.

Description and Context

Magma is a matrix organization split along functional and program lines, as is typical of a large bureaucratic organization. Automakers, aircraft manufacturers, defense contractors, and giant technology firms all exhibit the same characteristics. However, Magma's history reaches back to the beginning of the industrial age, and its legacy is an important part of the structure and culture. There have also been significant events in its organizational history that have established the tone of operations for decades into the future. Within this broader structure, I was under the direct supervision of a manufacturing systems engineering manager, which was 8 levels from the top of the organization. The chain of command is as follows:

1. CEO
2. COO
3. Executive Vice President of the Americas
4. Global Vice President of Manufacturing
5. Vice Presidents
    a. Vice President of Advanced and Manufacturing - VOME
    b. Vice President of VO Plants and Stamping
6. Director of Manufacturing Engineering
7. Chief Engineer
8. Manufacturing Systems Engineering Manager
9. Intern

The governing hierarchy and command chain in Magma is based on the functional group, which placed me solidly in the manufacturing engineering domain. From these functional groups, people are then assigned to vehicle programs to make up the other dimension of the matrix. These vehicle programs are also structured in a hierarchy to manage all of the required subsystems and functions for development.

Structural

Although Magma is nominally a matrix organization cut by function and vehicle programs, the current organization is clearly dominated by the functional arm. This domination is partially a consequence of applying moving assembly line techniques to product development in the early days of the company. When dealing with a complex system such as an automobile, the assembly is broken down into a variety of subsystems and their "constituent parts." Each worker is then assigned one task of the assembly to be repeated over and over, enabling a complete car to be assembled in minutes, rather than hours or days. This base assembly line technique has been refined over the ensuing decades to encompass lean manufacturing, one-piece flow, and manufacturing cells, but the core decomposition of a complex product into constituent parts remains.

When this principle is extended to the engineering and development processes, a similar logic applies. The complex system is broken into subsystems and smaller parts for engineering and design, and functions are created around the subsystems. Once these subsystems are designed, they are assembled in prototypes to figure out if the system is working properly. If the performance isn't up to specifications, the design process is iterated until the system works properly. However, there is no express method for feedback in the current intellectual design system before iterations occur. Most of the current feedback methods cover information transferred in the form of "lessons learned" or design reviews after deadlines are past. To be sure, the engineering tasks attempts to gather all of the relevant information for their part, but the information is not static. Overall, this process lends itself strongly to a functional and hierarchical layout, with the hierarchy managing the decomposition and reconstitution of the designs, and the functions executing the required tasks for each part. As Wallach diagrammed in his thesis, it is natural for a complex and stable product to require such an organizational system:

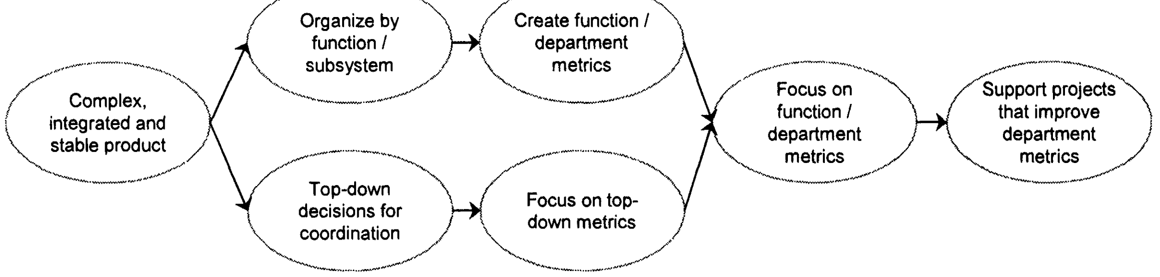

Complex, integrated and stable product
Organize by function / subsystem
Top-down decisions for coordination
Create function / department metrics
Focus on top-down metrics
Focus on function / department metrics
Support projects that improve department metrics

Figure 72: Casual diagram illustrating the derivation of a functional structure from a complex product (Wallach, *Solving a Corrosion Quality Issue at a Major US Automaker*)

However, a complex product by itself is not enough to spur the introduction of a massive hierarchy and functional breakdown. For example, there are a variety of small-volume car manufacturers that have much flatter organizations than Magma. These companies will typically employ smaller and more cross-functional teams of engineers to work on the subsystems for the one or two cars that they produce. Once a manufacturer begins to produce a variety of products that all require similar functional tasks to be executed in the design, there are perceived scale advantages to combining the functional experts and a functional organization emerges. This organizational structure then serves to complicate the ability to communicate across functions and ultimately introduces an assembly line technique to product development.

For much of its history, Magma's organizational structure was well suited to the key customer requirements. From the earliest days of the automobile up until just a few decades ago, the powertrain was the primary concerns of the customer. Powerful, lightweight, and reliable engines were the focus of development so that an automobile could be a reliable mode of transportation. For this reason, structuring the organization into its basic functional subsystems (body, chassis, engine, transmission, etc.) made sense because the groups of engineers required to develop a great subsystem for the customer were in the same place with very few barriers to communication. However, the ensuing 50 years of automobile development have perfected many of the previous customer requirements. Engines and transmissions are so reliable now that many people don't ever open their hoods themselves, and won't ever need to. Quite simply, the automobile has completely satisfied its basic need as a reliable form of transportation, leaving customers to focus on more refined attributes, such as fit & finish, aesthetics, interior design, and sensory perceptions of touch, sound, and sight. Unfortunately, many of these new customer requirements cross the existing functional boundaries. For example, dealing with squeaks and rattles or engineering a specific driving "feel" does not fall just within the engine group.

Said another way, functional decomposition of the automobile was much easier in the past because of its highly modular architecture. Over time, the product architecture has moved from modular to integral for a variety of reasons, including uni-body construction, aerodynamic improvements, and weight reduction efforts. In the terms of the preceding DSM analysis, these changes mean that the strength of the interactions has increased, thereby increasing the chance of iterations and rework. This newer integral architecture also requires a more integrated supply chain and organizational structure. For these reasons, Magma has essentially outgrown its current organization after mastering the previous customer requirements, and is now in a poor position to deal with the customer requirements of the broader system.

One consequence of the original approach to organization was the functional nature of the individual tasks in the subsystems, engineering work, and assembly. Engineers and assembler were unable to see far beyond their part, and certainly not beyond the subsystem, so specialization was valued highly. For this reason, Magma has long been staffed with incredibly talented and experienced engineers who could overcome detailed technical challenges. These technical achievements were then passed up the hierarchy for approval and combination with other subsystems. Unfortunately, approval in the hierarchy was a poor substitute for system engineering, since most practical system engineering then had to be done on the fly during the launch of a vehicle. Although this organization made for great subsystems, it left many of the broader system issues undiscovered until the vehicle was being produced for sale. In the past, this was not much of a problem since customers were more focused on basic subsystem reliability. However, today's customers are now focused on the very system features that Magma had been able to ignore for so long.

In some sense, delaying systems engineering and error discovery until launch is the most reasonable solution for this organization since all errors will be aggregated in one physical place on a prototype, and only those that affect customer requirements can be addressed. The alternative would be a serious and prolonged period of error discovery during the development process that would increase the number of iterations and exacerbate the existing problem with feedback delays. However, delaying error discovery until the prototype phase runs the risk of leaving significant latent flaws in the design launched to the customer. As Magma has famously discovered in its history twice before with vehicles it has produced, latent design flaws can cost the company billions of dollars in recall repairs and countless billions in lost goodwill, brand image, and future business. Consequently, every effort needs to be made to ensure that the hierarchical functional organization is not contributing to these design flaws. Because of the inherent communication difficulties between functional groups, the current organization is ill-suited to handling systems engineering without increased communication to reduce information feedback delays.

Internal Processes

Eventually, Magma documented its assembly line treatment of the product development process in the Magma Product Development System (MPDS). MPDS serves as the final roadmap for the development of any vehicle, all based on working backward from a set date for the production of the first car. Fundamentally, MPDS treats the development process as a set of constituent designs that need to be intellectually "assembled" into a final design. The estimated time required to complete each design is subtracted from the original deadline, gradually building a massive Gantt chart of milestones, deadlines, and required tasks that is used to structure, staff, and prioritize remaining work. This method of development planning makes MPDS the perfect embodiment of Magma's hierarchical and functional organization, assuming that no rework or coordination will be required unless mistakes are made. As a result, any systems engineering occurs within isolated "tech clubs" that bring together a variety of engineers to address a common problem. However, my brief experience with these "tech clubs" suggests that they are primarily used to address errors and issues, and do not represent a fundamental change in how the development process is conducted on a daily basis. When deadlines are missed due to required rework or unanticipated complications, the subsequent MPDS deadlines are extended and an acceleration plan is created. There is no concept of loops, feedback, or time delays that would cause MPDS to be inherently flawed. Rather, the entire structure of the organization is designed with faith in the assembly line breakdown of development work.

Very similar to MPDS, the Magma Manufacturing Design System (MMDS) is a current initiative to document the manufacturing process development tasks. MMDS is basically a flowchart representation of the manufacturing tasks involved in designed and finalizing a manufacturing process and plant layout for vehicle production. The use of flowcharts does allow for feedback in the documented processes, which gives MMDS a huge boost over MPDS in understanding the fundamental drivers of rework. However, the MMDS analysis is crippled by its relation to MPDS, largely because the flowcharts are all staged by existing MPDS milestones. At no point does the MMDS analysis step away from the structured assembly line process inherent in MPDS and analyze task relationships without an imposed order. As such, predetermined MPDS deadlines are baked into the MMDS analysis from the beginning, creating a preordained result that only documents the current procedures and does little to improve them. Ultimately, this reduces the MMDS initiative to a flowchart diagram of the current execution of functional tasks, and bolsters the current view of organizational structure.

However, this is not to say that Magma does not have a handle on its predicament. Rather, a complex initiative was launched in 1998 by Tony Zambito to understand the basic structure of the product development tasks. Using the same DSM tools applied in this thesis, Tony sought to modify the MPDS process by examining tasks and their interactions independent of the imposed deadlines. In the end, Zambito was able to collect a massive database of ~1000 tasks and all of their precedents and dependents, and then analyze it using the DSM framework. After the analysis was done, a new task order had been established, coupled blocks had been identified, and completion time distributions could be calculated. Much of the results spoke to the inter-related nature of product development tasks and the need to restructure the development process. However, for reasons that will be discussed in the political and cultural sections of the organizational analysis, Tony's structural work was eventually modified to fit into the existing MPDS framework by the upper management. False dependencies were added to force certain tasks into designated timeframes that corresponded with the current MPDS process. These false dependencies had the final result of bending the technical analysis to fit an old process and ultimately created no meaningful structural change in the organization or process.

As a result, Magma is still left with a development process that is fundamentally linear, like an assembly line. The central flaw in this assembly line breakdown is that the consequences of actions cannot be seen immediately. Also, the assumption of perfect decomposition implies that any work on the system level is a pure cost. Furthermore, rework is perceived to be a result of errors in the process, rather than a natural consequence of the time delays between executions of inter-dependent tasks. Each worker is focused on a single task or part and is fundamentally incapable of viewing the entire system. On a physical assembly line, this is not a huge problem because the time cycle of the process is relatively short. Q&A stations can see the overall system and can find, diagnose, and solve errors within a matter of a few production units or hours. However, product development time scales are much, much longer. When an engineer changes a design, its effect on the system may not be seen for months or until a prototype comes off the line.

Structural Dynamics

As a result of the long time delays for feedback, the core functional structure of Magma's development process will naturally restrict the ability to cross-coordinate and will ultimately hide errors in the system. Some of these errors will be found during the prototyping process, but a significant fraction will never be discovered or discovered after delivery to the customer. One potential solution to these hidden errors would be increasing cross-function coordination and communication to reduce the time delays for feedback. However, the intervening decades have strengthened Magma's original functional structure into today's rigid hierarchy. Unfortunately, this rigid hierarchy is ill-equipped to deal with reducing feedback delays because of the decreased coordination ability that comes with strong hierarchies and functional orientation. The following diagram summarizes the trade-off between functional strength and cross-coordination ability:

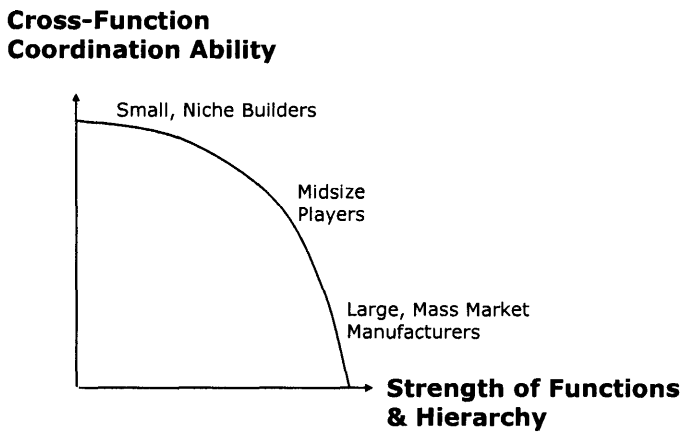

Cross-Function Coordination Ability
Small, Niche Builders
Midsize Players
Large, Mass Market Manufacturers
Strength of Functions & Hierarchy

Figure 73: Trade-off between functional strength and cross-coordination ability

In the absence of any compensating tools or practices to enhance communication across functions, increasing the strength of a functional hierarchy will reduce the ability to communicate across functional barriers. In Magma's case, this results in a scenario in which most cross-functional communication must go up one branch of the hierarchy, across, and then down the other side (commonly referred to as the "inverted U"). There are some notable exceptions with the "tech clubs" that get together to solve system problems, but the daily method of working is unchanged. Undoubtedly, this lack of coordination makes solving complicated systems problems harder, but it also reinforces an isolated and metric-driven structure. Because much of the organizational incentives are directed toward the isolated functions, Magma's hierarchical and functional structure does not lend itself to cross-functional coordination. However, there is another effect that complicates the loss of coordination ability:

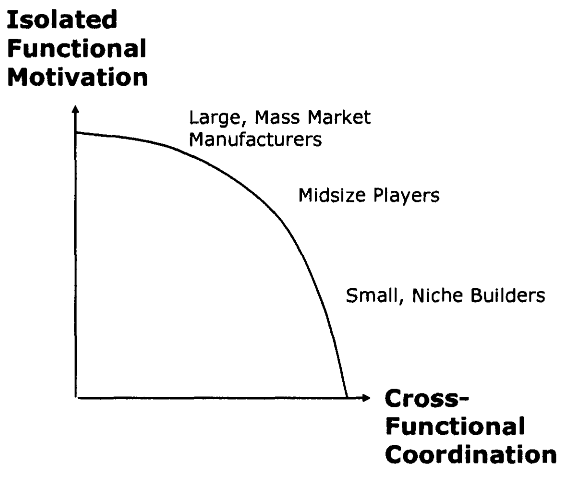

Isolated Functional Motivation
Large, Mass Market Manufacturers
Midsize Players
Small, Niche Builders
Cross-Functional Coordination

Figure 74: Trade-off between coordination and motivation

As the figure shows, there is also a trade-off between isolated functional motivation and cross-coordination within a company's functional groups. Essentially, increasing coordination between functional groups lowers the focus of isolated functions on their specific metrics or profit incentive. Rather, the increased coordination forces them to cooperate and meet a set of common goals, but this inevitably hurts the alignment of metrics and their work. In contrast, the absence of coordination drives individual motivation to achieve their local goal. For a plant or division, the relevant goal could be profit or cost per unit, but for individual employees or directors, the goal will be their individual performance metrics. Unless these metrics and incentives are perfectly aligned, each function will be ultimately working against its compatriots instead of toward a common system goal of customer satisfaction. Usually, this trade-off is used to highlight a coordination gap that developed when a functional organization is trying to motivate its groups with localized metrics.

Connecting the two concepts discussed above, it's simple to see that a matrix organization is inherently unstable. The following diagram illustrates the concept:

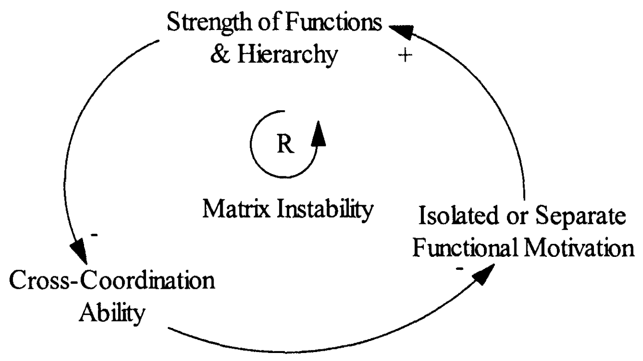

Strength of Functions & Hierarchy
+
R
Matrix Instability
Isolated or Separate Functional Motivation
-
-
Cross-Coordination Ability

Figure 75: Matrix instability from a reinforcing loop

As the image above shows, the two concepts generate a reinforcing loop that can pull a matrix organization in either direction in the absence of compensating effects. To illustrate, strengthening a hierarchy & functional orientation will reduce cross-coordination ability, which will increase individual functional motivation and consequently strengthen the hierarchy & functional orientation. In the reverse, weakening the hierarchy & functional orientation will increase cross-coordination ability, which will decrease individual functional motivation and consequently weaken the hierarchy & functional orientation. From these two dynamics, it's clear that matrix organizations are naturally an unstable equilibrium, and will tend to tip toward one side of the matrix unless management constantly strives to maintain the balance.

Currently, Magma's matrix organization is dominated by the functional side, suggesting that Magma has been traveling down this reinforcing loop for some time without management correction. As a result, Magma finds itself trapped in a functional orientation that is ill-suited to address systems engineering problems that bridge its functions. Fortunately, there are a few options for improvement. First of all, Magma could use the reinforcing loop in reverse to bring the organization back towards the middle of the matrix. Naturally, this would require a variety of measures to weaken the functional hierarchy and strengthen the cross-coordination emphasis. In addition, Magma could also strive to reduce the communication delays in the development process while still maintaining the deep technical expertise that comes from a functional organization. Graphically, this could be viewed as pushing out the frontier so that coordination ability isn't lost as functions gain strength, or as weakening the connection between strength of function and cross-coordination ability in the reinforcing loop. Finally, Magma could also weaken the connection between cross-coordination ability and isolated functional motivation by changing the local metrics. As a note, it is important for Magma to retain technical prowess, since technical experts are required for system engineering. Overall, there are several options for improvement:

1. Create a flatter organization with fewer barriers to cross-functional communication, similar to small-volume manufacturers
2. Create smaller program teams and work in team form
3. Create tools that can increase effective communication, or otherwise reduce time delays in feedback
4. Create dedicated system engineering function for internal coordination, similar to the closures system integrators

Although all of these solutions are intended to reduce feedback delay time or weaken the hierarchy & functional orientation, the implementation of a single one may not be enough to improve the development system. It is entirely possible that several solutions are necessary to get the required systems expertise, organizational structure, and set of tools to streamline the overall process. It is also vital to point out that the changes must make cross-functional communication happen quickly, reliably, and on a regular schedule – otherwise the tool may be deemed a failure.

One example of a solution back-firing is Magma's virtual series, which is a set of meetings and virtual analysis intended to provide a testing environment in which the assembly and fit of designed parts can be assessed before real parts are built. On paper, the virtual series is a wonderful idea, designed to provide cross-functional feedback to the engineering design teams on their work without building physical prototypes. However, the virtual series currently takes about 8 weeks to complete after data is frozen at a specific point, which fundamentally restricts the frequency of virtual series to at least 8 weeks. Meanwhile, engineering work continues and decisions are made without feedback from the virtual series. When the virtual series is finally completed, the engineers typically have to go back and rework decisions that were in the frozen data *and* the subsequent decisions that were made while the virtual series was progressing. In this manner, the 8 week duration of the virtual series actually introduces the very delays that it's supposed to prevent. Essentially, "working ahead" will result in additional rework. As such, any communication or system tools must be fast enough to be real-time or that engineering work can be stopped while waiting for the results. In many cases, the benefit of waiting and avoiding rework greatly outweighs the opportunity cost of idle labor while waiting, even if that time is 8 weeks. One potential solution for the virtual series is to focus on broad-based compatibility of assembly *before* addressing the final complexity. This would dramatically shorten the time to run a series.

Theoretically, there is an optimal frequency for a systems communication and analysis tool to be used. As stated above, the frequency cannot be too long because it either (1) creates rework if people continue working without appropriate feedback, or (2) costs too much in the lost opportunity of idle labor while people wait for the feedback. In addition, the frequency shouldn't be so short that no new meaningful analysis is generated from the cost of running the tool. For example, a virtual series run every hour would frequently generate no new results since it takes a number of days for engineers to produce changes. Ideally, the tool frequency would be a multiple of the fundamental time scale for completing changes and new work once feedback is received. In a computer coding environment, the time scale for changes and additions is on the order of hours, so the frequency for compiling the code (i.e., checking the system) is daily. For automotive development, the time scale for changes and additions is days, so the likely system tool frequency should be every few days or weekly. Regardless of the frequency of use, the systems tool should be able to process the input and generate feedback as quickly as possible. It doesn't make sense to have a tool take 8 weeks to generate output if the feedback is needed on a weekly basis. As such, the completion time must be less than the required frequency of use; otherwise, the system will oscillate out of control. These tools should then be built into the organizational structure as a bridge between functions.

Industry and Market Dynamics

To further complicate the difficulties with system engineering, Magma's aging organizational structure is also ill-suited to the company's changing strategy in the marketplace. The trend towards low-inventory, lean, and flexible manufacturing has prompted Magma to increasingly outsource more of its design work to suppliers, headlined by the spinning out of internal parts designers and producers a few years ago. However, the coordination-motivation framework can help to illuminate some of the resulting structural complications of outsourcing, using the spin-out as an example:

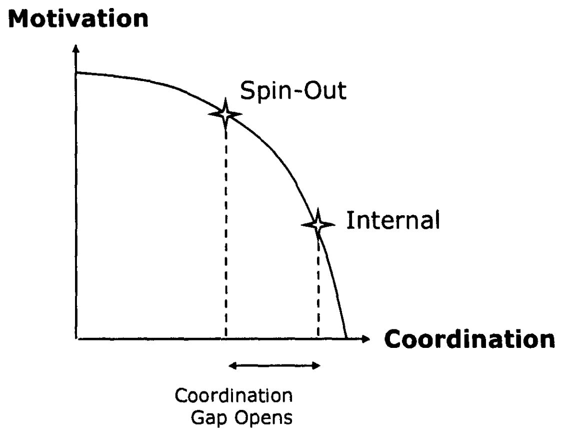


Figure 76: Coordination gap from spinning out an organization

As the two positions show, the hope of the spin-out was that the new firm would gain motivation as an independent company. With profit at stake as an independent supplier instead of internal Magma metrics, the hypothesis was that the spin-out would become more motivated to succeed and produce cost-competitive, high-quality parts as their customer set expanded. However, the increased motivation comes at the expense of coordination between entities. Internal part producers are generally easier to coordinate and manage than external ones, in spite of internal political dynamics, and especially when competing interests are at stake. In this particular case, Magma still has a dramatic structural advantage because it buys 90% of the spin-out's products. However, once the spin-out gains more independence, Magma will be unable to control or coordinate the relationship as well.

Ultimately, this leaves Magma with a coordination gap that must be addressed to be an efficient integrator and assembler of parts. There were a variety of solutions that could be employed to improve Magma's coordination ability mentioned above, and the principles governing these solutions should hold true for external coordination. Communication needs to show the results at the systems level quickly, reliably, and frequently. However, Magma needs to address the coordination gap or else it can expect systems engineering to get more difficult. One way to handle the increased external coordination with suppliers is creating a flatter hierarchy with responsibility pushed down to lower levels, but there are certainly other ways. Furthermore, it is unreasonable to ask a strictly functional hierarchy to handle systems coordination with suppliers, because it's unclear what function should hold that responsibility. Rather, Magma should make the vehicle programs a much stronger organizational feature throughout the company (i.e. bringing the matrix back toward the middle), thereby allowing cross-functional teams to be formed that manage a relationship with a number of suppliers to ensure that customer requirements are met for the subsystems. However, Magma needs to be careful with this shift of the matrix, since over-correcting will result in local optimums for each program cluster, thereby contributing to the divergence from any central standards. The suppliers themselves would then be forced to re-aggregate the various requests from Magma program teams into functional requirements for their own organizations. Program teams would also be responsible for sharing identified solutions for a set of supplier products between programs.

Without making this transition back toward the middle of the matrix, Magma will be caught halfway between business models. Their operations are no longer vertically integrated enough to support the aging functional hierarchy, and the organization has not transformed enough to support the new operational model. In fact, one of the only major organizational initiatives of the past few years has been the development of the Technical Maturity Model, which was a well-designed series of strategies intended to retain technical expertise in the company and encourage technical maturity on programs. Much of these maturity initiatives were designed to attack the standard 2-year promotional cycle, which often hampers a 4-year development cycle when talent running programs either (1) advances out of their program role, or (2) leaves the company for other opportunities. Overall, retaining this technical talent is vital to Magma's transition to a newer business model, because technical expertise is a required basis for systems integration. Without technical talent, Magma will end up having to outsource all design and will become just a brand name assembler providing little value to the customer. However, Magma does need to ensure that it also provides the systems tools that are required for technical experts to perform their design and integration roles properly. Without tools, the cross-functional coordination problems and supplier coordination gap will remain in place, creating delays in feedback and hampering the development process.

Throughout its history, Magma should have been spinning out systems that are no longer their core competency and restructuring the organization along the new functional criteria. If the original systems of the automobile were the engine, transmission, and chassis, a functional organization along those lines made a ton of sense. However, an organization that is addressing vague customer requirements such as interior feel, wind noise, closing effort, and drivability should bear little resemblance to the organization of old. Unnecessary functionality should have been spun out a long time ago, leaving Magma more streamlined to deal with the newer system requirements. If this had happened, it would have prevented much of the organizational growth that was required to design engines and manage the complexity of the combined systems into customer requirements. However, Magma has not considered any spin-outs until recently, thereby leaving itself with a massive organization of functional departments that has had significant trouble addressing customer requirements. Of course, Magma would argue that the situation is not an "either / or" decision, since quality subsystems *and* systems engineering are required to make a quality vehicle.

Magma should have been reassessing its core competency and deciding where to compete on a continual basis. The past fifty years have changed the marketplace to demand performance that Magma is no longer providing with its organizational structure. This lack of fit between organization and strategy is leaving Magma with a major decision as to whether they should be a subsystem supplier or an assembler, which is a question that blends competencies, capital investments, market size, and brand. In the end, the alignment between organization and strategy must be corrected before sustained profitability can be returned.

Political

As is natural, the political environment within Magma is tied to its organizational structure. A functional hierarchy obviously concentrates political power in the upper echelons of the organization and creates walls between functions. In addition, the vehicle development process dictates much of the information flow within the company, establishing Product Development as inputs to Vehicle Operations. This creates the feeling of one division being subservient to another, since VO must respond when the inputs from the PD change. Also, because Magma treats its development process as primarily linear, there is no real mechanism for the VO group to credibly feed information back into PD. To be fair, there are some mechanisms for engineering specifications and standards, but the power relationship is most certainly one-way. As a result, VO seems to be a permanent underling of PD. For these reasons, much of the political power in Magma is concentrated in the higher levels of the studio design and product design divisions.

The concentration of political power in these two locations has many detrimental effects on the organization's ability to function. First of all, it places the manufacturing and plant divisions at the back end of the process without the political power or organizational mechanisms to exert much influence over product decisions. As a result, the most capital-intensive area of the company is forced to respond to the product decisions. From a purely financial perspective, it is far cheaper to change a product to fit an existing manufacturing process than it is to change a process to fit a new product. Re-tooling a factory creates a financial burden that must be borne by the vehicle program and often constrains many of the subsequent decisions on the program regarding vehicle features and customer requirements.

Secondly, the concentration of political power in the top levels of the hierarchy cripples the organization's ability to solve system issues without traveling up the entire hierarchy. Because Magma's organization is structured along functional subsystems, resolving a systems problem often requires input and changes from a variety of divisions. However, the engineers who would normally agree on the changes to make to solve the issue do not have the power to make the changes. Rather, they have to go up a level to their supervisors, who then get together and discuss the solution, and then present it for approval to their supervisors. Ultimately, systems decisions must be reviewed by several levels of the hierarchy before they reach the subset of people (or the one person) that can actually authorize the change. Predictably, this process takes a long time to meander through the bureaucracy, and also results in frequent changes to the original decision as people review the data and leave their mark.

An excellent example of this dynamic occurs with the manufacturing and closures systems integrators (MSI's and CSI's) that are tasked with diagnosing and resolving systems issues such as wind noise and water leaks on doors. The SI's have frequently complained that their system improvements have been changed or reversed by the lead functional engineers on the programs, often leaving existing problems in place without a solution. Unfortunately, this leaves the SI's responsible for the systems issues but without any authority to enact the changes. As a result, the current state of "first among peers" places SI's in a purely advisory role and is inadequate for fighting against Magma's political hierarchy. However, it is not immediately clear why the SI's don't have any power until the broader political structure of Magma is examined.

Power of Finance

In spite of the concentration of political power in the upper level of the design divisions, much of the true power is held by the finance department. Magma revolves entirely around a set of financial controls and planning that are governed by the "black box" of the finance department. Finance is responsible for the allocation and forecasting of all resources, including placing engineers on vehicle programs, appropriating funds for capital improvements in plants, and setting profit goals for each program or major project. However, many people will admit to not understanding how finance arrives at its decisions or models for allocation of resources. As a result, everyone in the functional divisions is beholden to a set of rules that often seem arbitrary and contrary to system improvement. In reality, many of the financial allocations come down to the political power wielded by a vice president. In the case of the system integrators discussed above, there was a dramatic difference in the push given by each VP in charge. As a result, the CSI's had much more structure and resource authority than the MSI's did.

To complicate matters more, the MSI's have no separate budget for their activities. Rather, each MSI is still employed and paid by their "home" functional group, and is viewed as being "on leave" to work on systems issues. Even worse, the supervisor in charge of their system integration work has no reporting power over them. This creates a hold-up situation in which an old functional boss can either call a MSI back to work on their old tasks, can threaten the MSI if decisions do not help that particular function, or can simply bias the MSI's against other functions. Obviously, this frequently places the MSI's in an awkward position balancing the desires of their system integration supervisor versus the demands of their home functional department. In many ways, this situation is also true for the CSI's, since they still report to their home product clusters, as well. A separate budget would go a long way to overcoming these power issues within Magma, but would simultaneously hamper the elimination of walls between functions with the act of erecting another one.

Regardless, the lack of a budget or accountability for systems improvement renders any CSI-type initiative essentially powerless within Magma. A systems budget would not only allow for influence within the hierarchy, but it would also mean accountability for the executive management. As the initiatives currently stand, there seems to be no real plan of execution, no performance metrics, and no accountability above first level supervisors. Participants are left to influence, cajole, and reason with their peers and superiors to achieve the goal of system optimization. Since this method of convincing is in stark contrast to how things normally take place within the Magma hierarchy, systems initiatives typically stumble along for a while and then slowly fade away. From talking to many people about past initiatives, it is clear that most participants view these system initiatives as a side project that, no matter how useful, is a distraction from the real work to be done. Even worse, it's clear that failure has no consequences. There are no jobs to lose since everyone is still paid by their functional departments, there is no money to lose because no budget exists, and there is no promotion to forfeit. Everyone can walk away from these initiatives no worse for the wear, aside from the opportunity cost of lost time. In a sense, this attempt at cross-coordination does not break the link to isolated functional motivation shown in Figure 75, and the matrix instability continues.

Even worse, Magma maintains no central budget for large scale capital or strategic improvements. Nominally, this budget doesn't exist because it would be the first budget to be pillaged by a company with a history of dramatic swings in profitability. However, its non-existence means that complex initiatives that would benefit every vehicle program are required to be funded by individual programs, and to be within their profit targets. Predictably, this means that system wide initiatives are never even proposed, because they simply can't be sustained by a single program alone. In fact, these profit targets are so rigid that vehicle programs are often held to them even if dramatic changes in assumptions are required in the middle of the program. One solution to these financial controls would be making system wide initiatives their own cost centers (or profit centers), or very least their own budgets. The primary issue is that the positive externalities and benefits to scale are not always seen or valued by the individual entities within Magma. This would allow Magma to generate benefits that come from unifying practices, and could make a justifiable business case for the billions of dollars of investment required to update the facilities. As it currently stands, no vehicle program is profitable enough to undertake a billion dollar investment, and certainly has no incentive to do so when the benefits would not be completely captured by that single program.

A further example of the financial controls addresses the transition of a vehicle program from development to production. During the development of the vehicle, the program covers the product and process specifications with the primary deliverable of a producible vehicle that meets all of the specifications. At this stage, the vehicle program has a budget that is designed to cover the development costs to reach launch. However, once the development and initial launch tests are complete, the vehicle program budget is passed on to the plant for production. Subsequent production costs, including warranties and recalls, are then borne by the plant. This transition of budgets removes a dramatic incentive to discover and address system errors, primarily because the vehicle program that would discover these engineering errors will *never* see the costs of their errors. This is a classic time-lag problem of externalities from economics, correctly predicting that the vehicle programs will ignore problems that other entities bear. In short, the vehicle programs are "polluting" the plant budgets with extra costs from hidden errors. Normally, this issue would be correctable in the organization through the metrics, but the inherent political relationship between the programs and plants will need to be addressed before simply changing the metrics.

Political Change

Eventually, if Magma credibly changes its structure to match the changing nature of the system issues that it must resolve to meet customer requirements, then many of these political problems will disappear. Although structures cannot be guaranteed to dictate the political environment, a significant enough change will prompt a shift in the politics. As addressed above, a new structure will have to incorporate several features to encourage and speed feedback between the major functions, if not changing the definition of functions themselves. If the concepts of feedback and system engineering are introduced to the development process, it will be increasingly difficult for the upstream functions to hold power over the downstream functions simply through the input-output relationship. The entire development process will become much more circular through the feedback processes, which should even the playing field dramatically. In addition, Magma needs to push much of the responsibility for systems decisions down into the ranks of engineers and not require the decisions to travel the entire height of the hierarchy.

Reorganizing to empower systems decisions should not only relieve the upper management of much of the burden of managing the decision process, but also reduce the concentration of political power in the top. Finally, an improved systems structure would also reduce the power of the finance department by establishing more general metrics for evaluation that do not so heavily on individual budgets, cost per unit, and allocations. Single plant, program, and division profit or cost targets breed political infighting and poor system decisions through actions that benefit the individual but not the group.

Unfortunately, it would be exceedingly rare for the current holders of power to willingly abandon their positions for the good of the company. Change is usually resisted, and changing the power structure is always resisted by those destined to loose influence in the new structure. Rather, these decisions will need to be forced into place as part of a massive revitalization designed to structure Magma around the customer requirements that need to be perfected in today's cars. In the end, there are two stages on the path out of the political quagmire in which Magma finds itself. As discussed above, an immediate solution would be giving budget authority to the systems integration function. This would give a sense of legitimacy and power to the systems initiatives, prompting true involvement and accountability for the changes. However, this tactic largely uses the current organizational structure against itself. Creating a separate "systems" function with a separate budget plays right into the current functional structure and could ultimate hurt cross-functional communication by enforcing organizational walls instead of bridging them. So, a longer term solution to the political problems must focus on changing the power structure both by reorganizing the functions as discussed in the structural section, and by shifting towards power through ideas and merit instead of hierarchy and pure authority.

Cultural

The culture of an organization is exceedingly difficult to divorce from the inherent structural and political environment, especially when the historical context is taken into account. Organizations are fundamentally a group of people that have evolved over time, thereby establishing a self-fulfilling state of existence that is the product of intensely inter-dynamic behavior. However, a static view of the current culture is vital to diagnosing an organization's problems and how best to address them. Magma's current employee culture is best described by the following three points:

1. Operates on an extremely short term horizon
2. Defers to hierarchy and rejects operating outside traditional channels
3. Feels a sense of fatalism

Short Term Outlook

First of all, it has been clear in all of our observations that the vast majority of work at Magma is done on an extremely short time horizon, ranging from launch teams solving immediate daily problems to a broader focus on metrics for the current quarter or program. This behavior can largely be traced to the MPDS timeline, the financial metrics for the functional groups, and the promotional cycle. As mentioned above, the MPDS timeline for development is set by establishing the date for Job #1 and backing up a series of deadlines to the project start date. As a result, the experience on a development program becomes a series of short term deadlines and requirements that are designed to stage completion of the tasks, but actually produce a feeling of constant stress. Quite frequently, work is hurried to meet these deadlines, often producing errors that will be fixed by launch teams later, or simply creating work after the deadline. Obviously, this adds to an already stressful culture and enhances a feeling of completion after a deadline is passed. In reality, a deadline may be just the beginning of a set of iterations designed to produce a final converged design, but people are loathe to return to a design after a deadline has been passed. As a consequence, much of the rework is deferred until the launch team can handle it.

The financial metrics complicate this short term view by establishing incentives for localized and short term optimization. For example, a vehicle program will typically have specific cost per unit metrics to keep the overall price of an automobile under a certain threshold set by the marketing and finance departments. These cost per unit metrics are often sacred, and many decisions are made based on whether or not the metric will be violated. Culturally, this breeds an incredibly cost-conscious culture that will often reject investment opportunities because the initial cost (which must be borne by an individual program) will hurt in the metric in the short term in spite of long run benefits. In addition, these metrics bias Magma towards certain solutions, such as heavy automation and the use of robots, which have low cost per unit metrics during production. However, the automated solutions that have the lowest cost per unit are typically not the most flexible solutions, creating very high setup costs for the next program to use the facility. In the end, the financial metrics block a longer term view of flexible investments that would help multiple programs over time, or even encourage the use of people over robots.

Finally, the 2-year promotion cycle set within a 4-year development cycle creates a self serving interest for individual. When the rotation of personnel is faster than the cycle for one program, employees end up leaving programs in the middle of development. Obviously, this means that program knowledge is walking out the door, leaving the replacements to carry on the original path. Predictably, there is a significant ramp-up period for new employees coming onto an existing program, and the reasons for previous decisions are often not understood completely. Continuity becomes increasingly difficult to maintain, especially without any formal method of education or training for the program. Unfortunately, the replacements are then tempted to make their own mark on the program in their 2-year stint, often reversing or modifying previous decisions and thereby extending the development time. This temptation is completely natural, because each new person is being evaluated on their impact and performance, not their flawless continuation of someone else's work. However, their behavior simply enhances the short term view throughout Magma.

Magma has recognized the problem and has been doing some things to address the 2-year promotion cycle. The technical maturity model (TMM) is a new promotion track designed to retain engineering staff on a program, providing advancements and greater responsibility within a given project. Hopefully, as responsibility on the project increases, the longer term view of the development will be enhanced. However, Magma needs to be very careful that rewarding functional and technical expertise does not compromise the broader goal of systems integration by strengthening the metrics for individual success in one functional area.

Deferential to Hierarchy

In an operational sense, Magma is definitely ruled by its hierarchy. Decisions cannot be made without your supervisor's approval or by following the official process. Quite frequently in my initial time here, my requests for services or interviews would be met with the question, "Who's your supervisor?" or "Does your supervisor know what you're doing?" However, when that information was provided, everything was smoothed over. It became clear that I had no accountability and that my supervisor was viewed as ultimately responsible. In talking to others throughout the organization, it was clear that my experience was not unique. This transference of responsibility to supervisors seems to carry all the way up to the VP's, and was often compared to a military organization. In fact, much of the current culture was attributed to the leadership of the Whiz Kids in the 1950's and their institution of many military practices and strict financial controls, although the contributions of the founder are without comparison. As a result, even today's employees tend to be incredibly deferential to the hierarchy and the established ways of doing business. Many seem to think that this is the correct way to run a disciplined organization, but many others seem resigned to the fact that fighting the hierarchy is useless and damaging to their careers.

As a result, the established system is rarely questioned in any real form. Many of the engineers that I interviewed understood and agreed with my identification of the core issues at Magma, but none seemed ready to stake themselves against the system. Obviously, part of this reaction is due to the history of failed initiatives at Magma and the inherent complexity of the system, but a large part seemed to be trapped by the hierarchy. Individually, they didn't have the power or authority to make the change, and traveling all the way up the hierarchy would be too hard. Some engineers even seemed fearful of the vice presidents, saying that whenever the higher-ups were involved it was because something had gone wrong. Magma's culture is also highly resistant to those operating outside the traditional, or approved, channels. The clearest example of this behavior lies in the reputation of supervisors who have access to the higher levels of the organization. These supervisors are often viewed as suspicious because of their connections, and the ultimate effect undermines the change initiative in spite of an attempt to get the issues in front of an audience with enough authority to actually change the system.

Magma further enhances the feeling of separation and hierarchy with a variety of perquisites and status items that apply as one is promoted up the chain. Most notably, the addition of assistants serves to create a significant barrier between supervisors and managers, especially when the assistants are used to restrict access to the decision makers. In the basement of the PD building, the restriction actually used to be physical. Every manager's assistant sat between the main office and the hallway, creating a physical barrier to access. Today, the barriers seem to be virtual, largely because the assistants handling the blocking over email. Obviously, none of this is new – assistants have long been the key to the decision makers in organizations – but there is little feeling of an open door policy on any level. Designated parking spots, executive conference rooms, washrooms, office size, and provided (and serviced) cars all serve to enforce the hierarchy and a broader feeling of separation between the levels.

Sense of Fatalism and Depression

Throughout my conversations regarding the systems initiatives, I was unable to bring about a true sense of enthusiasm from anyone. In spite of their understanding of the core problems, many people still communicated the feeling that trying to change was useless, and that they felt trapped in the organization. Some simply felt trapped by the massive organization around them, largely because they had witnessed many attempts at change and none had succeeded. Essentially, they were left with a bitter and jaded feeling toward fighting the organizational inertia built up over decades. At this point, any new ideas to fix Magma are viewed as fads and aren't trusted. Others were trapped because of a lack of skills or wage transferability, in that they would prefer to leave Magma but would be unable to find a similar paying job in Detroit and were unwilling to move their families. All felt secure in the fact that Magma had been around for 100 years and probably wasn't going anywhere. Ultimately, these feelings created a focus on job security and meeting their individual performance metrics, obviously hampering the attempts at systems integration. This sense of fatalism even extended to the issue of bankruptcy, in which one employee voiced the belief that "the federal government would never let us fail – there are too many jobs at stake." After witnessing the previous federal bail-outs of industry, the belief is certainly well founded. The promise of a bail-out creates a massive safety net for Magma and removes many of the incentives to improve, thus helping to perpetuate the status quo.

However, the ultimate effect of this attitude has been quite strange. In spite of their rather obvious organizational depression, most employees still feel a large measure of pride in their work. They are clearly able to separate their efforts from the troubles of the whole system, largely with the perception that they're doing a good job even if others are messing things up.

Naturally, this breeds a culture of blame and complicates the issue of cross-functional communication. Even worse, these feelings create a pervasive sense of detachment from the problems at hand. Employees do not feel involved in the overall product, even if their efforts are central to the outcome. As a result, Magma seems to be in some sort of organizational denial when they should be in a crisis mode. One hypothesis is that this is some sort of response to the continuous stress of trying to catch the competition. The reality is that Magma has been dealing with their core problems for years, and no one can sustain a sense of urgency for that long. Rather, employees have become accustomed to the stress and constantly live under it; they're desensitized and untrusting, much like anyone under continuous stress.

Obviously, the path to this cultural state has taken decades to travel. Much of the trouble began with the changes that the Whiz Kids put in place in the 1950's to institute a financially controlled, metric driven, functional hierarchy. However, the history of Magma has served to exacerbate many of the problems while the marketplace simultaneously moved to render their organization and culture unfit for the required strategy. Naturally, the reversal of these trends will take years, if not decades, to rectify. The path out to a healthy culture must include a variety of changes, covering new system-focused metrics, a less rigid hierarchy, greater responsibility and accountability in the lower ranks, and an easing of the jaded feeling. However, this will be a bit of a trick for Magma to complete. Traditionally, companies lacking a sense of urgency need to have one stilled to motivate employees toward the new goals. In Magma's case, employees seem so jaded that no threat of change is credible. Perhaps only the threat of bankruptcy and job loss would be enough to force change within Magma, but those scenarios are currently voided by the tacit promise of a government bail-out.

One possible tactic is to use the employee culture against itself for some time. A formal process for organizational change could be instituted that is similar to the change process used in engineering design. In engineering, there are a variety of systems used to identify, track, and solve design flaws from the bottom up. If these systems are followed, most of the relevant issues will be found and solved with the appropriate people involved. Given Magma's faith in the hierarchy and established ways of business, it's possible that the creation of a formal organizational change process, when combined with the requisite backing from top management, could actually force change in the organization through an established channel. Unfortunately, the complex nature of the organization does not lend itself to a solution built from the bottom up. Regardless, it seems unlikely that this would be a long term solution to the cultural issues. Magma truly needs to move its culture to be more collaborative and idea-based in order to identify and meet new customer requirements for vehicles. This is probably only possible when changes in the organizational structure are coupled with employee turnover for decades.

Synthesis and the Framework for Change

In spite of the value in separating the structural, political, and cultural elements of an organization, these three pieces are irrevocably linked through an organization's history. When establishing a company, the organizational structure is typically set first to match a given strategy and the type of people being hired to execute the strategy. The structure and character of people then establishes the political power relationships, and sets the company in motion. Over time, the organization evolves into an entity that perpetuates its structure, politics, and culture through employee turnover and hiring practices. People are hired that will work well in the given environment, and the culture ingrains itself. Without outside influence and correction, the organization will tend to become increasingly homogeneous and set in its ways. In this manner, Magma has evolved its original functional structure and technical culture into a massive hierarchy that emphasizes individual metrics and authority.

However, the demands of the marketplace often change faster than an organization's ability to adapt. Even worse, the time spent in changing the culture can be far longer than changing the politics, and longer still than changing the structure. As such, Magma finds itself in a position where its strategy and organization do not fit, and with a long time lag to change the organization. So, the logical questions become where to begin the change process and what is the best way to do it? The overall time lag of change makes matching strategy and organization a moving target, but it's also unclear whether change should begin with the longest time scale element (culture), or the shortest (structure). Unfortunately, the answers to these questions are not immediately clear, but Jan Klein's framework for change will help provide some direction

(Klein, *True Change: How Outsiders on the Inside Make Things Happen in Organization).* Klein categorizes the relationships within an organization along the following three dimensions:

Basis for legitimacy:

Technocratic $\Longleftrightarrow$ Experience Based

Basis for relationships:

Lateral $\Longleftrightarrow$ Hierarchical

Basis for support:

Merit $\Longleftrightarrow$ Authorization

Figure 77: Framework for organizational change (Klein, *True Change: How Outsiders on the Inside Make Things Happen in Organization)*

These three dimensions are used to categorize and predict the approach of change that will be most effective inside an organization. Legitimacy is split between technocratic organizations that value technical expertise and qualifications versus those that value experience in the company or industry. Relationships are broken laterally across functions and vertically through the hierarchy. Finally, the basis for support is split between the merit of an idea, often supported by data, and the authority to force change.

In Magma's case, the organization is on the right side of all three dimensions. Experience is clearly valued, with many people proud of their 25 year histories in the automotive industry, and experience being a fundamental prerequisite for a leadership position. Unfortunately, installing such experienced people will usually tend to perpetuate existing ways of doing business and make the organization risk averse. Magma's relationships are also extremely hierarchical, with only a few places in which lateral relationships are encouraged. Obviously, this is part of the reason why the broader system issues with vehicles are difficult to resolve. Finally, Magma's basis for support is almost exclusively authority. Because it is an engineering firm, data is often required to justify and prioritize the merit of decisions, but change will not happen without authority. Because of the gating effect of authority, Magma must be categorized on the right side of the spectrum.

Ultimately, an organization that resides on the far right side of the three dimensions must enact change through a conduit that (1) has been in the industry for a long time, (2) resides in the top levels of the hierarchy, and (3) has the authority to reach all parties. Essentially, Magma needs someone to step in and take the reins to force change down from the top. In such as organization, the real job of a change agent is pushing from the top, seeding people at the bottom, breaking a resistant middle, and generally creating an environment for change. This leaves very few people left in the organization that can undertake changes of the magnitude addressed in this thesis. If Magma wants to change its vehicle development processes to design products to the manufacturing processes instead of the reverse, then the only person left to do that is the CEO. A change of that magnitude covers every major function and requires a dramatic influx of resources. Lesser changes, such as creating system tools, can be instituted by the VP's, but changing the entire development process is probably the job of the CEO.

Interestingly, Klein's framework is primarily intended to enact change within a given organization. If processes need to change or system tools need to be installed, her framework provides an excellent guideline. But how does one work to change the entire organization itself? Some of the major recommendations thus far have been to make Magma less experience-based, less hierarchical, and less authoritative. Much of their current problems are a result of these inherent features of the organization, and it seems hypocritical to permanently use authority to force someone to be more merit-based, or to use the hierarchy to force lateral relationships. Consequently, it seems that the core elements must be used against the organization at first, but then more permanent changes must be made in the organization itself. Overall, the tactic should be similar to my initial suggestion to give budget authority to a systems function as a first step to a truly systems-oriented organization, which would ultimately have an entirely different structure. As a result, Klein's framework seems to be a starting point for change *within* an organization by "working the system," but needs to be extended with longer term solutions that address the fundamental structural, political, and cultural elements that determine categorization along the dimensions. Essentially, it's only possible to use an organization against itself for so long before the organization must evolve into its new form. Ultimately, the framework is merely the basis for the huge changes that are required to realign the fit of an organization and its chosen strategy in the marketplace.

Gap Analysis

Unfortunately for the prospect of change, there is nearly always a significant gap between an organization's view of its problem and the root causes. The following diagram illustrates the gap and its place in the change process:

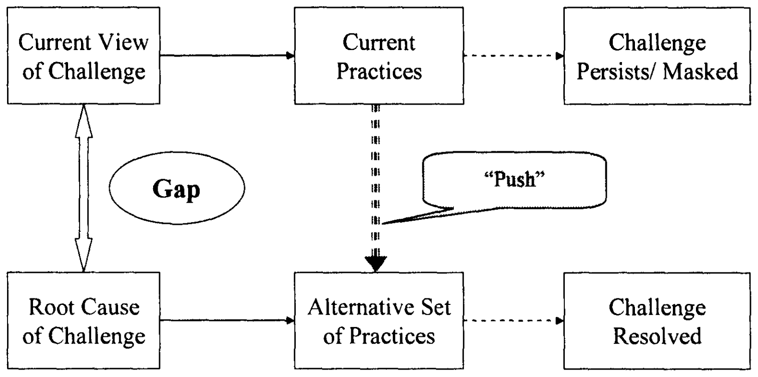

Current View of Challenge
Current Practices
Challenge Persists/ Masked
Gap
"Push"
Root Cause of Challenge
Alternative Set of Practices
Challenge Resolved

Figure 78: Gap in understanding challenges (Klein, *True Change: How Outsiders on the Inside Make Things Happen in Organization*)

As the figure shows, change often occurs when someone pushes a new set of practices or ideals upon others. Often, this pushing creates substantial resistance and hampers the effort to improve. In reality, it is much better to find someone on the inside who sees the situation from an outside perspective to "pull" in the change. These "outsider-insiders" are frequently much more effective at pulling change into the organization, although some amount of pushing will almost always be necessary. However, these outsider-insiders must first be able to identify the gap between the current view of the problem (or the current mental models) and the true cause of the problems. If no one is able to see the gap or the root cause, then the system will continue to perpetuate itself and its consequent problems.

In Magma's case, there is still a significant gap in the middle of the organization between the current view of the problem and the root cause, and this gap seems to be relatively steady in spite of some major initiatives designed to push change through Magma. There are five major reasons for this sustained gap:

1. Many believe that Magma is close to the optimum and that mistakes are the only problem
2. Organization is extremely risk averse
3. Most feel constrained by the system
4. Few outsider-insiders exist within Magma, and those present are not empowered
5. No one can to see complete system

One of the most widely held beliefs in Magma, especially in upper management, is that the current problems with vehicle development are execution issues. As a VP and director both said, "People are making mistakes – if we just did things right the first time, then none of these problems would happen." Of course, they're right – people are making mistakes and creating errors that remain hidden until the vehicle is launched. However, their assumptions about the mistakes are misguided. The mistakes do not derive totally from human error, as they assume, but from the inherent structure and task relationships in the development process. These mistakes are unavoidable, and there is no way to "do things right the first time."

Consequently, most people at Magma naturally assume that they are close to the optimum already, and all that's needed to be the best vehicle manufacturer is to properly execute their process. Of course, Magma is close to *an* optimum because it's always possible to optimize any given process. However, the analysis suggests that Magma's current process happens to be a local optimum, not the global one. Even if Magma recognizes that it's currently at a local optimum, there is little chance that the leadership would attempt the level of investment required to make the move to a better state. Quite simply, the organization is too risk-averse and too fearful of radical change. Billions of dollars would be required to revamp the manufacturing plants, processes, structures, and metrics over the ensuing decades, and there's no assurance that Magma would be able to reach the global optimum. In fact, there is a very real fear that beginning the change and only making it halfway through would spell the downfall of the company. Consequently, it seems that much of the leadership believes that it's not sensible to risk a functioning system that covers thousands of employees and billions of spent capital in the pursuit of lost potential. In addition, risking the system would undoubtedly make Magma worse before it gets better. The transition is not an immediate one and profit, not to mention individual performance metrics, would suffer heavily before the situation improved. As a result, many people seem to view change as career limiting because things are bound to get worse before they get better. Overall, the feeling is that the inertia of the current system is so high so that Magma is "too big to fall" and that risking the entire system is a poor decision.

Beyond the fear of change and risk aversion, many would-be leaders also seem to feel constrained by the system. Functional separations, a strong hierarchy, and individual performance metrics all tend to limit actions to those approved by the formal processes, which are not well suited to change. In addition, Magma also lacks a formal change process for organizational change, thereby robbing any would-be change agents from a formal channel that would be automatically accepted by the organization. As a result, the complex system that Magma has designed to execute their development and manufacturing processes ultimately constrains any attempt at change. So, any change process must begin with a set of outsider-insiders that truly care about the organization and are in a position to pull in change. These outsider-insiders will have to work both from the bottom up and the top down in order to change Magma's culture and formal structure at the same time.

Unfortunately, there is a severe lack of empowered (or visible) outsider-insiders at Magma, and Magma's organization is not set up to encourage the development and growth of a base set of these people. First of all, it is tough to retain an outside perspective at Magma because of the experienced-based, hierarchical, and authoritative nature of the organization. Anyone entering the organization as an outsider isn't viewed as credible because they don't have experience in the company, even if they have experience in the industry. In addition, it is simply not possible to rise within the hierarchy without becoming an insider and conforming to the established processes. As a consequence, very few people have been able to hold positions in the top of the hierarchy as an outsider-insider for a long time. Most people either give up their outsider status, go "underground" and bide their time, or depart the company in frustration. This leaves Magma without a critical mass of outsider-insiders who are able to comprehend the problems and use available opportunities to pull in changes. Even worse, these dynamics create a split in hierarchy between the outsider-insiders and the pure insiders. Those outsider-insiders that are still left in the organization tend to be in the middle or bottom of the organization where they can see the problems but are left to be frustrated and constrained by the system above them. In contrast, the leaders in the top of the organization tend to be insiders or outsiders that are hiding their perspective. Obviously, this is the reverse of the situation that one would hope for in an organization that needs to change. A far better situation would be allowing the lower levels to focus on the quick wins and "low hanging fruit" in the operational details of the work, while the top levels are able to see the bigger system and enact significant reforms.

Even if Magma did have a critical mass of outsider-insiders, there is an additional factor that inhibits their ability to change. Outsider-insiders are most effective pulling in change when they are in the "right place at the right time," or otherwise able to take advantage of the challenges within the company. One way to describe the opportunities is through the following diagram:

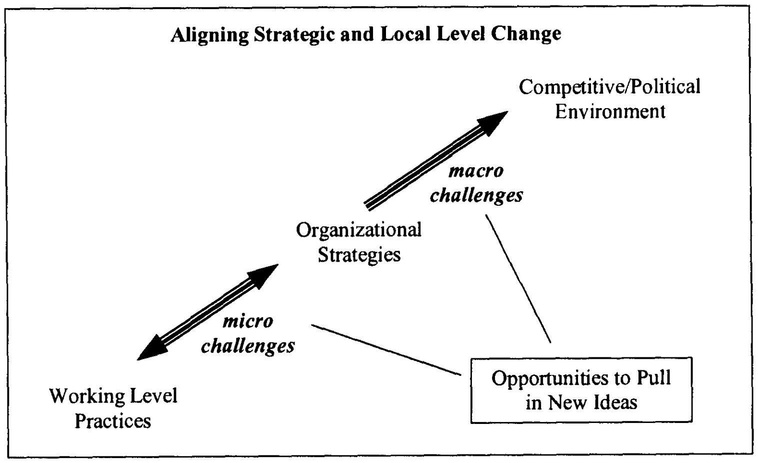

Aligning Strategic and Local Level Change
Competitive/Political Environment
macro challenges
Organizational Strategies
micro challenges
Working Level Practices
Opportunities to Pull in New Ideas

Figure 79: Opportunities for change arise from new strategies (Klein, *True Change: How Outsiders on the Inside Make Things Happen in Organization*)

As the figure shows, most of the opportunities for change come from broader challenges presented by shifting strategies. These opportunities are essentially macro and micro mismatches between the organization and its new strategy. On a macro level, the functional hierarchy may no longer be the correct structure for a company's new competitive environment, or the political situation may be spiraling out of control. On the micro level, the working practices and operational details may no longer be adequate for the quality demands of the marketplace. In either scenario, these mismatches present challenges that outsider-insiders can seize to bring in their experiences, views, and ideas.

On the surface, Magma is in a perfect position to execute change in the face of these macro and micro challenges. They are currently in the midst of a massive mismatch between their overall corporate strategy and their organizational structure. Magma is ripe for outsider-insiders to take the reins and pull in change. However, change isn't happening, because of the fundamental inability to see the complete system. It is virtually impossible for one person to get their head around something as complicated as developing and manufacturing a vehicle. The system is so complex that we have to break the actions up into thousands of individual tasks, frequently neglecting many of the important interactions when those tasks are reassembled. In addition, the functional nature of Magma's processes prevents many people from even having the access to all of the required data, much less the ability to process it all. As a result, there are a limited number of people that have been able to take a complete view of product development and process development activities across the entire firm.

This inability to view the entire system is one of the reasons why the results in this thesis are bound to be so controversial. When a set of tasks spanning product development and process development are analyzed together, the results suggest that there is a very high chance that the system is non-convergent. This mathematical result does make some intuitive sense, and can predict many of the problems that are experienced in real vehicle programs. However, this obvious mathematical result is not obvious organizationally. It does not make sense that Magma could have survived for so long with a fundamentally flawed development process, but that is exactly what has happened. Even worse, there is substantial evidence that better systems exist in the Japanese automobile manufacturers, who had the luxury of beginning their systems well after Magma was already locked into its processes. The development of an organization is highly path dependent, and Magma simply started off on the wrong foot for today's customer requirements. As a result, changing the organization to meet today's competitive demands will be exceedingly difficult for them, even if they were once perfectly aligned with the demands of the marketplace.

Plan for the Future

Fortunately, all is not lost for Magma, although they are likely to have a tremendous amount of difficulty in moving their organization forward. The simultaneous mismatch of the broader strategy and organization, when coupled with the lack of a critical mass of outsider-insiders, makes sustaining any change process incredibly difficult. Really, the only reliable way to produce change is starting from both the bottom and top by obtaining high level backing for some pilot programs to demonstrate quick improvement. However, Magma needs to ensure that the change progresses into the middle of the company as well. The following four steps can help to pull and push change throughout the entire organization.

1. Secure more high level outsider-insiders
2. Use pilot programs for proof of concept
3. Tie promotion and career advancement to change initiatives
4. Encourage the development of outsider-insiders

The first step in the change process has to be securing more outsider-insiders in the top levels of the hierarchy. As noted in the framework for change, this person is vital to advancing change in the experience-based, hierarchical, and authoritative Magma organization. Furthermore, the changes required in Magma are of such magnitude that it would impossible to begin without some access to resources and high-level approval. Securing a high-level supporter also enables the process to begin from both the top and bottom. The high-level supporter can work the system by using the hierarchy to force change throughout the system, but can also enable changes in strategies that allow people at the bottom to pull in change. One of these changes should be the use of pilot programs to test use development strategies on vehicle programs.

First and foremost, using pilot programs to demonstrate the proof of concept should provide a ton of data to justify the risk in moving all of the operations over to process-centric design. Pilot programs should also provide a series of small wins that can be used to extend the change initiative. In fact, Magma has already demonstrated that the new development processes work on the smaller, niche-market vehicle prototypes. Both of these vehicles were developed by small cross-functional teams working closely together and designed to the manufacturing processes that would be used to build them. Unfortunately, there is a huge catch to the success of both of these pilot programs. Both of the cars are built by hand and represent a much more flexible manufacturing process than would be normally available to a product development team. Magma needs to introduce some pilot programs that develop vehicles according to the processes already contained in one of their plants. From conversations with some of the manufacturing engineers, roughly 40% of Magma's current plants have processes that are worth retaining. One of these plants should be chosen for a pilot program for vehicle development, using the GPDS development schedule as a guiding influence. GPDS is currently an outsider in the organization, and it would behoove Magma to bring GPDS to the forefront. Of course, the pilot programs do not have to begin with the design of a new car. Rather, the pilot programs should start with product line freshenings and progress up the development schedule in complexity as they are deemed successful.

Of course, the use of pilot programs should not be confined just to the technical side of the business. Magma would also do well to use pilot programs to extend the proof of concept to organizational dynamics. With the proper construction, a pilot program could not only serve to improve the product, but also the working relationships within the companies. In some ways, the existing CSI program is a good example because it represents a significant departure from the traditional way of doing business at Magma. Thus, even if the ultimate product is only marginally improved, there is a good chance that Magma will still be able to use the CSI program to illustrate the benefits of a more lateral and merit-based organization.

As the pilot programs grow in size, it is vital to tie the advancement of people to their success on the pilot programs. If there is no accountability or reward for participating in the programs, Magma will be clearly communicating that change is merely a side project. Even worse, if the individual functional performance metrics aren't changed, pulling change into the organization will still be viewed as detrimental to career advancement. Obviously, this would not encourage the advancement of outsider-insiders in the organization. Ultimately, Magma should be angling to establish itself as an organization that is permanently able to adapt its organization to meet the changing strategies in the marketplace. The only way to do this is by building a set of outsider-insiders throughout the organization that are able to pull in change when the challenges arise. If

Magma does not encourage the development of these people, then the company will be forced to always push change down from the top.

Although beginning the change process from the top and bottom creates the greatest chance for success in the changing the vehicle development processes, the timetable for the pilot programs will be decades. It remains an open question whether Magma wants to take years to slowly create larger and larger pilot programs that justify process-centric design, especially when there is a significant chance that the marketplace will change again before their transition is complete. For example, a new fuel type could return the engine and transmission functions to the forefront. Magma will also be trying to standardize its parts and processes during this change period, and it may prove difficult to do that while pilot programs are running that may inform the future standards. As a result, it may make more sense for Magma to invest in pushing change in one fell swoop so that the effort can be completed sooner. However, the faster the change, the more expensive and disruptive it will be to the current process. If Magma decides to move to standardized flexible manufacturing processes in all of its plants and adjust their development process accordingly, the cost will run into the billions of dollars and take more than a few years to complete. In the meantime, Magma is exposing itself to all of the risks that it would like to avoid in moving away from their local optimum and towards the global one.

V. Conclusion

In spite of the vast range of potential benefits to redesigning their development processes using DSM tools, Magma appears to be destined for a slow rate of change. The sheer size of the organization, coupled with the difficulty in agreeing upon standards and communicating new directions, virtually ensures that the change management process will take decades to fully complete. However, the problems have also had decades to ingrain themselves, ranging from the initial application of linear assembly line techniques to product development to the rigid financial and staffing controls applied in the 1950's. All combined, undertaking the transformation of product and process development will be met with decreased performance before improvements are felt. In other words, Magma will definitely get worse before it gets better, thereby giving the competition ample time to attack during the transition. However, this should truly be viewed as investment in the future, as a time in which Magma can cede unprofitable market share, use positive cash flows to invest in new processes, and emerge as a much stronger competitor with an organization matched to its corporate strategy.

More generally, the lessons at Magma illustrate the problems any company can face in developing rigidities. As Magma grew over time, its organization became highly specialized for a given set of tasks in developing and producing vehicles. However, this specialization has proven to be a double-edged sword by significantly restricting Magma's ability to adjust to changing market demands. As such, the vast majority of the conclusions and recommendations in the organizational analysis are directly applicable to any industry or firm that has developed rigidities. Although a framework has been built for choosing the best way to break an organization out of its rigid system, little work has been done on whether an organization *should* change or not. From an efficiency perspective, it is entirely possible that the changes required to remake Magma are simply too costly. Essentially, this would mean that the market prefers Magma to be shut down, especially because struggling to save a firm could represent a significant principle-agent problem. For example, most workers at Magma would be interested in saving their jobs and livelihoods, although their action represents a lower return for the shareholders. However, this perspective neglects the short run costs that the economy must bear in lost jobs, destroyed brand image, etc. Future research should be directed away from the question of how to change a company in favor of addressing why. When are the costs of change greater than the benefits? Should externalities be included in the decision? When is it better to release assets to the market versus being retained in the existing firm? Answering these questions will help to identify situations in which firms should be shut down instead of being dragged along in a less optimal state.

VI. Table of Figures:

VII. Table of Equations:

VIII. References:

Browning, Tyson R., *Modeling and Analyzing Cost, Schedule, and Performance in Complex System Product Development,* MIT Thesis, 1998.

Browning, Tyson R., and Eppinger, Steven D., "Modeling Impact of Process Architecture on Cost and Schedule Risk in Product Development," *IEEE Transactions on Engineering Management,* Vol. 49, 4: 428-442, November 2002.

Eppinger, Steven D., Nukala, Murthy V., and Whitney, Daniel E., "Generalised Models of Design Iteration Using Signal Flow Graphs," *Research in Engineering Design,* 9: 112-123, 1997.

Eppinger, Steven D., and Salminen, Vesa, "Patterns of Product Development Interactions," *International Conference on Engineering Design,* ICED 01 Glasgow, August 21-23, 2001.

Eppinger, Steven D., Whitney, Daniel E., Smith, Robert P., and Gebala, David A., "A Model-Based Method for Organizing Tasks in Product Development," *Research in Engineering Design,* 6: 1-13, 1994.

Ford, David N. and Sobek, Durward K., "Explaining the Second Toyota Paradox by Modeling Real Options in Product Development," Submission to *IEEE Transactions on Engineering Management,* February 2004

Kanter, Rosabeth, *Confidence: How Winning Streaks and Losing Streaks Begin and End,* Crown Press, 2004.

Klein, Janice, *True Change: How Outsiders on the Inside Make Things Happen in Organization,* Chapters 1-4, Jossey-Bass, 2004.

Repenning, N., P. Goncalves, and L. Black (2001). "Past the Tipping Point: The Persistence of Fire Fighting in Product Development," *California Management Review*, 43, 4: 44-63.

Repenning, N. (2001). "Understanding Fire Fighting in New Product Development," *Journal of Product Innovation Management*, 18, 5: 285-300.

Smith, Robert P., and Eppinger, Steven D., "A Predictive Model of Sequential Iteration in Engineering Design," *Management Science,* Vol. 43, 8: 1104-1120, August 1997.

Smith, Robert P. and Eppinger, Steven D., "Identifying Controlling Features of Engineering Design Iteration," *Management Science,* Vol. 43, 3: 276-293, March 1997.

Teixeira, Marcelo C.M., Marchesi, Henrique F., and Assuncao, Edvaldo, "Signal Flow Graphs: Direct Method of Reduction and MATLAB Implementation," *IEEE Transactions on Education,* Vol. 44, 2: 185-189, May 2001.

Thomas, Robert J., *What Machines Can't Do: Politics and Technology in the Industrial Enterprise,* University of California Press, 1994.

Wallach, Jeremy, *Solving a Corrosion Quality Issue at a Major US Automaker,* MIT Thesis, 2004.

http://www.dsmweb.org

IX. Appendix 1: Top 50 Eigenvalue Tears

Task	Precendent	Resulting Eigenvalue
Prog Strategy - Theme - Design Concepts Defined	Product & Process Feasibility	0.78275
Product & Process Feasibility	Develop Program Specific Bill of Process	0.77577
Product & Process Feasibility	Manufacturing Targets	0.76852
Product & Process Feasibility	Manufacturing Consensus Review	0.76267
Product & Process Feasibility	Preliminary Process Design	0.75645
Product & Process Feasibility	Process Design	0.75075
Assembly - Process - Flow And Tolerances Defined	Develop Program Specific Bill of Process	0.73223
Manufacturing Targets	Virtual Series	0.72313
Product & Process Feasibility	Mfg - SM - Process - Pdpd Reqmts Defined	0.71651
Manufacturing Plan	Virtual Series	0.71088
Confirm Manufacturing Plan	Virtual Series	0.70848
Implement Manufacturing Plan	Virtual Series	0.70819
Theme - Ext - Single Theme Selection	Product & Process Feasibility	0.70205
Manufacturing Targets	Develop Program Specific Bill of Process	0.69323
Cutlines - Frt Door - Cutlines Defined	Product & Process Feasibility	0.68779
Develop Program Specific Bill of Resource	Develop Program Specific Bill of Process	0.6826
Theme - Ext - Theme Selection For Market Research	Product & Process Feasibility	0.67787
Preliminary Process Design	Develop Program Specific Bill of Process	0.67259
Welding - Weld Patterns, Types, Quality, Class Defined	Develop Program Specific Bill of Process	0.66876
Welding - Carryover/surrogate Weld Data Defined	Develop Program Specific Bill of Process	0.66595
Theme - Single Theme Compatible With Primary Pkg Reqmts	Product & Process Feasibility	0.66129
Gage & Monitoring Equipment Development	Plant Layout	0.65912
VTTO and Code X/Y Material Budget Forecast	Plant Layout	0.65707
Tooling Design	Plant Layout	0.65526
Theme - Ext - Multiple Themes Defined	Product & Process Feasibility	0.65096
Reuse - Carryover Parts In PDM	Develop Program Specific Bill of Process	0.64837
Long Lead Tooling and Facility Procurement	Plant Layout	0.64742
Confirm Plant Layout	Plant Layout	0.64679
Supplier Sourcing Plan	Plant Layout	0.64617
Labor Optimization	Develop Program Specific Bill of Process	0.64385
Water Mgt - Doors - Effective Drainage Compliant	Product & Process Feasibility	0.64096
Material Flow Plan	Develop Program Specific Bill of Process	0.6388
Feasibility Checkpoint #0	Develop Program Specific Bill of Process	0.63725
Develop Specific Virtual Assembly Simulations	Develop Program Specific Bill of Process	0.63588
Develop Discrete Event Simulation Models	Develop Program Specific Bill of Process	0.63535
Locators - Electrical Locators Defined In PDM	Develop Program Specific Bill of Process	0.63485
Safety Assessment	Product & Process Feasibility	0.63256
Product / Process Capability Target	Product & Process Feasibility	0.63026
Cutlines - Rr Door - Cutlines Defined	Product & Process Feasibility	0.62837
Locators - Frt End Sm Locators Defined In PDM	Develop Program Specific Bill of Process	0.62786
MDS #1	Product & Process Feasibility	0.62657
Material - SM - Spec And Thickness Frozen	Product & Process Feasibility	0.62544
Develop Future State Plant Layout	Develop Program Specific Bill of Process	0.62496
Locators - Closures Locators Defined In PDM	Develop Program Specific Bill of Process	0.62451
Locators - Soft Trim Locators Defined In PDM	Develop Program Specific Bill of Process	0.6241
Conduct Formal Target Estimation	Product & Process Feasibility	0.62311
Feasibility Analysis Summary Report	Product & Process Feasibility	0.6221
Material - Non SM - Spec And Thickness Frozen	Product & Process Feasibility	0.62144
Locators - Body Shell Locators Defined In PDM	Develop Program Specific Bill of Process	0.62105
Locators - Hard Trim Locators Defined In PDM	Develop Program Specific Bill of Process	0.62069

X. Appendix 2: Code

Ordering Matrices

dsmorder.m

```
% Primary ordering routine for the DSM
function [frontorder,midorder,backorder,order]=dsmorder2(AVG)

% create order row at bottom of matrix
n=size(AVG,2);
for i=1:n
   AVG(n+1,i)=i;
end

% Initialize variables and clear diagonal of matrix
frontplace=1;
backplace=n;
DSM=AVG;
for i=1:n
  DSM(i,i)=0;
end

fronts=1;
backs=1;
mids=1;
frontorder=[];
midorder=[];
backorder=[];

%Beginning of main ordering loop
while frontplace<=backplace

%Move empty rows
totalmoves=0;
while totalmoves~=-1
rowsum=sum(DSM,2);
moves=0;
for z=1:n
  if rowsum(z)==0
    DSM=swaptasks(DSM,moves+1,z);
    for i=z:-1:moves+3
      DSM=swaptasks(DSM,i-1,i);
    end
    moves=moves+1;
  end
```

```
end
totalmoves=totalmoves+moves;
if moves>0
  frontorder(fronts,totalmoves-moves+1:totalmoves)=DSM(n+1,1:moves);
  fronts=fronts+1;
  order(totalmoves-moves+frontplace:totalmoves-1+frontplace)=DSM(n+1,1:moves);
  DSM=DSM(moves+1:end,moves+1:end);
  n=size(DSM,2);
else
  frontplace=frontplace+totalmoves;
  totalmoves=-1;
end
end

%Move empty columns
totalmoves=0;
while totalmoves~=-1
colsum=sum(DSM(1:end-1,:));
moves=0;
for z=1:n
  if colsum(z)==0
    DSM=swaptasks(DSM,z-moves,n);
    for i=z-moves:n-2
      DSM=swaptasks(DSM,i,i+1);
    end
     moves=moves+1;
  end
end
totalmoves=totalmoves+moves;
if moves>0
  backorder(backs,n-moves+1:n)=DSM(n+1,n-moves+1:n);
  backs=backs+1;
  order(backplace-moves+1:backplace)=DSM(n+1,n-moves+1:n);
  backplace=backplace-moves;
  DSM([n+1,n-moves+1],:)=DSM([n-moves+1,n+1],:);
  DSM=DSM(1:n-moves+1,1:n-moves);
  n=size(DSM,2);
else
  totalmoves=-1;
end
end

%Reachability matrix to compare intersection to row set - first level only
midcount=0;
midflag(1:n)=0;
reach=DSM(1:n,:);
```

```
for i=1:n
  reach(i,i)=1;
end
reach=reach^n;
for z=1:n
  a=reach(z,:)>0;
  b=(reach(:,z)>0)';
  if a.*b==a
    midflag=a; %Match all precedents and dependents, and write out to establish levels
    break
  end
end

%Move midorder to front and separate for swapping
moves=0;
for z=1:n
  if midflag(z)==1
    DSM=swaptasks(DSM,moves+1,z);
    for i=z:-1:moves+3
      DSM=swaptasks(DSM,i-1,i);
    end
    moves=moves+1;
  end
end

%Establish base time calculations for midorder block
MID=DSM(1:moves,1:moves);
MID(moves+1,1:moves)=DSM(n+1,1:moves);
MIDTEMP=MID;
for i=1:moves
    MIDTEMP(i,i)=1;
end
midseq=seq(ones(1,moves),ones(1,moves),MIDTEMP);

%Begin swapping loop for midorder segment
bestswap=0;
while bestswap ~= -1
clear distances;
distances(1:moves,1:moves)=0;
for i=1:moves
  for j=i+1:moves
    if MID(i,j)>0
      distances(i,j)=MID(i,j)*(sin(pi/4-atan(i/j))*(i^2+j^2)^.5)^2;
    end
  end
end
```

```
bestdistance=sum(distances(:));
[imax,jmax]=find(distances==max(distances(:)));
jmax=max(jmax);

bestswap=-1;
for z=1:moves
  if z~=jmax
    MID=swaptasks(MID,z,jmax);
    distances(1:moves,1:moves)=0;
    for i=1:moves
      for j=i+1:moves
        if MID(i,j)>0
          distances(i,j)=MID(i,j)*(sin(pi/4-atan(i/j))*(i^2+j^2)^.5)^2;
        end
      end
    end
    if sum(distances(:))<bestdistance
      bestswap=z;
    end
    MID=swaptasks(MID,z,jmax);
  end
end

if bestswap~=-1
  MID=swaptasks(MID,bestswap,jmax);
end
end
%End of swapping loop

%Store values
MIDTEMP=MID;
for i=1:moves
    MIDTEMP(i,i)=1;
end

if seq(ones(1,moves),ones(1,moves),MIDTEMP)<=midseq
    midorder(mids,1:moves)=MID(moves+1,1:moves);
else
    midorder(mids,1:moves)=DSM(n+1,1:moves);
end

mids=mids+1;
order(frontplace:frontplace+moves-1)=midorder(mids-1,1:moves);
frontplace=frontplace+moves;
```

```
DSM=DSM(moves+1:end,moves+1:end);
n=size(DSM,2);

end
%End of ordering loop
```

dsmordernoswap.m

```
function [frontorder,midorder,backorder,order]=dsmorder2(AVG)

n=size(AVG,2);
for i=1:n
   AVG(n+1,i)=i;
end

%for i=1:size(best,1)
%   AVG(best(i,1),best(i,2))=0;
%   SD(best(i,1),best(i,2))=0;
%end

frontplace=1;
backplace=n;
DSM=AVG;
for i=1:n
  DSM(i,i)=0;
end

fronts=1;
backs=1;
mids=1;
frontorder=[];
midorder=[];
backorder=[];

while frontplace<=backplace %Beginning of main ordering loop

%Move empty rows
totalmoves=0;
while totalmoves~=-1
rowsum=sum(DSM,2);
moves=0;
for z=1:n
  if rowsum(z)==0
    DSM=swaptasks(DSM,moves+1,z);
    for i=z:-1:moves+3
      DSM=swaptasks(DSM,i-1,i);
    end
    moves=moves+1;
  end
end
totalmoves=totalmoves+moves;
if moves>0
  frontorder(fronts,totalmoves-moves+1:totalmoves)=DSM(n+1,1:moves);
  fronts=fronts+1;
  order(totalmoves-moves+frontplace:totalmoves-1+frontplace)=DSM(n+1,1:moves);
  DSM=DSM(moves+1:end,moves+1:end);
  n=size(DSM,2);
else
  frontplace=frontplace+totalmoves;
  totalmoves=-1;
  %fronts=fronts+1;
end
end

%Move empty columns
totalmoves=0;
while totalmoves~=-1
colsum=sum(DSM(1:end-1,:));
moves=0;
for z=1:n
  if colsum(z)==0
    DSM=swaptasks(DSM,z-moves,n);
    for i=z-moves:n-2
      DSM=swaptasks(DSM,i,i+1);
    end
     moves=moves+1;
  end
end
totalmoves=totalmoves+moves;
if moves>0
  backorder(backs,n-moves+1:n)=DSM(n+1,n-moves+1:n);
  backs=backs+1;
  order(backplace-moves+1:backplace)=DSM(n+1,n-moves+1:n);
  backplace=backplace-moves;
  DSM([n+1,n-moves+1],:)=DSM([n-moves+1,n+1],:);
  DSM=DSM(1:n-moves+1,1:n-moves);
  n=size(DSM,2);
else
  totalmoves=-1;
  %backs=backs+1;
end
end
```

```
%Reachability matrix to compare intersection to row set - first level only
midcount=0;
midflag(1:n)=0;
reach=DSM(1:n,:);
for i=1:n
 reach(i,i)=1;
end
reach=reach^n; %Could have infinite number problem here - fix to ones
for z=1:n
 a=reach(z,:)>0;
 b=(reach(:,z)>0)';
 if a.*b==a
   midflag=a; %this version breaks after finding the first match - pulling it and its dependents /
precedents - this does not identify "levels" correctly (could be 2 level 1's and it will pull the first)
but it will result in being able to identify the smaller subloops with a larger block by placing
them on separate rows in the midorder array
   break
 end
end

%Move midorder to front and separate for swapping
moves=0;
for z=1:n
 if midflag(z)==1
   DSM=swaptasks(DSM,moves+1,z);
   for i=z:-1:moves+3
     DSM=swaptasks(DSM,i-1,i);
   end
   moves=moves+1;
 end
end

MID=DSM(1:moves,1:moves);
MID(moves+1,1:moves)=DSM(n+1,1:moves);

midorder(mids,1:moves)=MID(moves+1,1:moves);

mids=mids+1;
order(frontplace:frontplace+moves-1)=midorder(mids-1,1:moves);
frontplace=frontplace+moves;

DSM=DSM(moves+1:end,moves+1:end);
n=size(DSM,2);

end %End of ordering loop
```

Timing Calculations

para.m

```
% Parallel time calculation based on Eppinger / Smith paper
function [time]=para(A)

n=size(A,2);
A=A(1:n,:);

for i=1:n
  u(i,1)=A(i,i);
  A(i,i)=0;
end

[S,Lambda]=eig(A);

if rcond(S)<1e-25 %if can't be inverted, send to get dominant eigenvalue only
    time=para2(u,A);
else
    M=(eye(n)-Lambda)\eye(n); %safer to calculate the inverse using G-J method than an inverse
call.
    N=S\eye(n);
    U=S*M*N*u;
    time=abs(max(U));
end
```

seq.m

```
% Sequential timing calculation based on Eppinger /Smith paper
function [time]=seq(DSM)

n=size(DSM,2);
DSM=DSM(1:n,:);

for i=1:n
  for j=1:n
    if (i==j)
      P(i,j)=1;
      B(i,1)=DSM(i,j);
    else
      P(i,j)=-DSM(j,i);
    end
  end
end
```

```
[L,U]=lu(P);
D=diag(diag(U));
M=D\eye(n);
N=L\eye(n);
x=M*N*B;
time=sum(x);
```

signalflow.m

```
%signal flow graphs
function [avg,var]=seq4(DSM)

n=size(DSM,2);
DSM=DSM(1:n,:);

times=diag(DSM);
times=[times;0];

for i=1:n
  DSM(i,i)=0;
end

adder=0;

%initialize and fill starting values
Net(1,:)=[1 1 2 1 times(1)];
Coeff_Names={[num2str(1) '*z^' num2str(times(1))]};
NetRow=2;
a=[2 3];

%build all the loops - average must match seq(A)
for n=1:size(DSM,2)
   A=DSM(1:n,1:n);
   A(n+1,:)=ones(1,n)-sum(A);
   for z=1:n
      if (A(n+1,z)<0)
         A(n+1,z)=0;
      end
   end
   [i,j]=find(A~=0);
   i=i+ones(size(i,1),1);
   j=j+ones(size(i,1),1);
   A=[0 a;a' A zeros(n+1,1)];
   for z=1:size(i,1)
      Net(NetRow,:)=[NetRow A(1,j(z)) A(i(z),1) A(i(z),j(z)) times(i(z)-1)];
      Coeff_Names{NetRow}=[num2str(A(i(z),j(z))) '*z^' num2str(times(i(z)-1))];
      NetRow=NetRow+1;
   end
   b(1:size(a,2))=(a(1,end)+size(a,2)-1):-1:a(1,end);
   b(1,end+1)=a(1,end)+size(a,2);
   a=b;
end

   [n,d]=mason2(Net(:,1:3),Coeff_Names,1,max(Net(:,3)));
   L=diff(simple(sym(n)/sym(d)));
   z=sym('z');
   M=diff(z*L)-L^2;
   z=1;
   avg=eval(L);
   var=eval(M);
```

Reducing Matrices

dsmcombine.m

```
% Script to combine the parallel blocks into single tasks

%Rebuild AVG in the new order and reorder SD
%DSM=builddsm(AVG,SD);
n=size(DSM,2);
for i=1:4
   DSM(n+1,:,i)=[1:n];
end

DSM=DSM([order n+1],[order],:);
LC=LC([order],1);
ResourcesRequired=ResourcesRequired([order],:);
%SD(1:n,1:n)=SD([order],[order]);
NAMES=NAMES([order],:);
FINAL=NAMES;

%Begin collapse of matrix using combine routine
for i=1:size(frontorder,1)
  if sum(frontorder(i,:)>0)>1

[DSM,LC,ResourcesRequired,NAMES]=combine2(DSM,LC,ResourcesRequired,frontorder(i,[fi
nd(frontorder(i,:)>0)]),4,NAMES);
  end
end
for i=1:size(midorder,1)
```

```
  if sum(midorder(i,:)>0)>1

[DSM,LC,ResourcesRequired,NAMES]=combine2(DSM,LC,ResourcesRequired,midorder(i,[fin
d(midorder(i,:)>0)]),3,NAMES);
  end
end
for i=1:size(backorder,1)
  if sum(backorder(i,:)>0)>1

[DSM,LC,ResourcesRequired,NAMES]=combine2(DSM,LC,ResourcesRequired,backorder(i,[fi
nd(backorder(i,:)>0)]),4,NAMES);
  end
end
```

combine.m

```
% Function to perform the mathematics of combining the DSM
function
[DSM,LC,ResourcesRequired,NAMES]=combine(DSM,LC,ResourcesRequired,order,type,NA
MES);

start=find(DSM(size(DSM,1),:,1)==order(1));
stop=find(DSM(size(DSM,1),:,1)==order(end));

n=stop-start+1;
SubDSM(1:n,1:n,:)=DSM(start:stop,start:stop,:);
SubLC(1:n,1)=LC(start:stop,1);

% Calculate times and resources
if type==3 %calculated blocks - simulated
  [WorkTimes,WorkPath,newavg,newsd]=dsmsimulate(SubDSM,SubLC,5,5,[1:n]); %keep
original suborder in simulation
  %store as lognormal!!! minimum value is stored as imaginary portion of complex SD to avoid
another DSM construct
  mu=mean(WorkTimes-min(WorkTimes)); %convert to lognormal parameters
  sigma=std(WorkTimes-min(WorkTimes));
  newavg=log(mu^2/(mu^2+sigma^2)^0.5);
  newsd=(log((sigma/mu)^2+1))^0.5+min(WorkTimes)*i; %store as complex...
  csvwrite('wt.csv',WorkTimes);
  %newavg=para(SubDSM(:,:,1));
  %newsd=0;
  %[newavg,newvar]=signalflow(SubAVG);
else %shortcut method for front and back calculations (no dependencies) - parallel times and
resources
  newavg=max(diag(SubDSM(:,:,1)));
  i=find(diag(SubDSM(:,:,1))==max(diag(SubDSM(:,:,1))));
  d=diag(SubDSM(:,:,4));
  newsd=max(d(i)); %use max sd of multiple max avg...
end

%Resource combination
NAMES(start,3)=real(sum(NAMES(start:stop,2).*NAMES(start:stop,3)))/real(max(NAMES(sta
rt:stop,2)));
for j=1:size(ResourcesRequired,2)

%ResourcesRequired(start,j)=real(sum(NAMES(start:stop,2).*ResourcesRequired(start:stop,j)))/
real(max(NAMES(start:stop,2)));
   %ResourcesRequired(start,j)=max(ResourcesRequired(start:stop,j));

ResourcesRequired(start,j)=max(real(sum(NAMES(start:stop,2).*ResourcesRequired(start:stop,j
)))/newavg,max(ResourcesRequired(start:stop,j)));
end

%LC Combination - weighted average by durations...
LC(start,1)=sum(LC(start:stop,1).*NAMES(start:stop,2))/sum(NAMES(start:stop,2));

for i=start:stop
   for j=1:4
      DSM(i,i,j)=0;
   end
end

% the actual combination
NAMES(start,5)=max(abs(eig(DSM(start:stop,start:stop,1))));

%separate out panes for processing later...
%SD=DSM(1:end-1,:,4);

% this works for panes 1-3, ok for pane for in triangular distribution (but not normal!) - straight
average of all non-zeros
DSM=[DSM(:,[1:start-1],:) sum(DSM(:,start:stop,:),2)./max(1,sum(DSM(:,start:stop,:)>0,2))
DSM(:,[stop+1:end],:)];
DSM=[DSM([1:start-1],:,:); sum(DSM(start:stop,:,:),1)./max(1,sum(DSM(start:stop,:,:)>0,1));
DSM([stop+1:end],:,:)];
DSM(start,start,1)=newavg;
DSM(start,start,2)=0;
DSM(start,start,3)=0;
DSM(start,start,4)=newsd;
DSM(size(DSM,1),start,:)=order(1);

% this is for pane 4 if normal distribution
%SD=[SD(:,[1:start-1]) sum(SD(:,start:stop).^2,2).^0.5 SD(:,[stop+1:end])];
```

```
%SD=[SD([1:start-1],:); sum(SD(start:stop,:).^2,1).^0.5; SD([stop+1:end],:)];
%SD(start,start)=newsd;

%replace SD pane...
%DSM(1:end-1,:,4)=SD;

%resource combination
NAMES(start,2)=newavg;
NAMES(start,4)=type;
NAMES=[NAMES(1:start,:); NAMES(stop+1:end,:)];
ResourcesRequired=[ResourcesRequired(1:start,:); ResourcesRequired(stop+1:end,:)];

%LC combination
LC=[LC(1:start,:); LC(stop+1:end,:)];
```

Eigenvalue Tearing

Dsmoptimize.m

```
buildvar;
%[AVG,SD,NAMES]=buildraw(input('Interaction Value? '));
%[frontorder,midorder,backorder,order]=dsmorder(AVG,[]);
threshold=input('Threshold Eigenvalue? ');

%besttasks=csvread('besttasks.csv');
%for k=1:size(besttasks,1);
%   buildvar;
%threshold=besttasks(k,3)+.01;
Best=[];
BestTasks=[];
i=1;
Lmax=0;

%determine if midorder segments need to be rehashed to meet threshold
while i<=size(midorder,1)
 if sum(midorder(i,:)>0)>1
  midtemp=midorder(i,[find(midorder(i,:)>0)]);
  SubAVG=AVG(midtemp,midtemp);

  for j=1:size(Best,1)
     if not(isempty(find(midtemp==Best(j,1))))
        SubAVG(find(midtemp==Best(j,1)),find(midtemp==Best(j,2)))=0;
     end
  end
```

```
    for j=1:size(SubAVG,2)
      SubAVG(j,j)=0;
    end

    if max(eig(SubAVG))>threshold
      [SubAVG,best]=dsmtear(SubAVG,threshold);
      for j=1:size(best,1)
        Best=[Best;midtemp(best(j,1)) midtemp(best(j,2)) best(j,3)];
        BestTasks=[BestTasks;NAMES(midtemp(best(j,1)),1) NAMES(midtemp(best(j,2)),1)
best(j,3)];
      end
      [forder,morder,border,oorder]=dsmorder(SubAVG);

      start=find(order==midtemp(1,1));
      stop=find(order==midtemp(1,length(midtemp)));
      order(start:stop)=midtemp(oorder);

      midorder(i,[find(midorder(i,:)>0)])=0;

      for j=1:size(forder,1)
        n=size(forder,2);
        frontorder(end+1,[find(forder(j,:)>0)])=midtemp([forder(j,[find(forder(j,:)>0)])]);
      end
      for j=1:size(morder,1)
        n=size(morder,2);
        midorder(end+1,[find(morder(j,:)>0)])=midtemp([morder(j,[find(morder(j,:)>0)])]);
      end
      for j=1:size(border,1)
        n=size(border,2);
        backorder(end+1,[find(border(j,:)>0)])=midtemp([border(j,[find(border(j,:)>0)])]);
      end

    end

  end
  i=i+1
end

dsmmode
dsmcombine
%Modes(1:size(Mode,1),2*k-1:2*k)=Mode;
%Eigs(1,2*k-1)=Eig;
%clear Mode
%clear Eig
%end
```

dsmtear.m

```
function [SubAVG,best]=dsmtear(SubAVG,threshold);

n=size(SubAVG,2);
SubAVG=SubAVG(1:n,1:n);

remove=1;

for i=1:n
   SubAVG(i,i)=0;
end

Lmin=max(real(eig(SubAVG)));

while Lmin > threshold

[i,j]=find(SubAVG<1 & SubAVG>0);
%[i,j]=find(triu(SubAVG)<1 & triu(SubAVG)>0); %if ordered going in

%[S,L]=eigs(SubAVG,1);
%Lmin=L;
%Smax=max(real(S));

for z=1:length(i)
   Temp=SubAVG(i(z),j(z));
   SubAVG(i(z),j(z))=0;

   L=max(real(eig(SubAVG)));
   Ls(z,remove)=L;
   if L<Lmin
      Lmin=L;
      bestindex=z
   end
   SubAVG(i(z),j(z))=Temp;
end

best(remove,1)=i(bestindex);
best(remove,2)=j(bestindex);
best(remove,3)=Lmin;
SubAVG(best(remove,1),best(remove,2))=0
remove=remove+1

end
```

dsmmode.m

```
%kill interactions in Best
for i=1:size(Best,1)
   AVG(Best(i,1),Best(i,2))=0;
   SD(Best(i,1),Best(i,2))=0;
   %AVG(find(order==Best(i,1)),find(order==Best(i,2)))=0;
   %SD(find(order==Best(i,1)),find(order==Best(i,2)))=0;
end

%Pull out maximum eigenvalue and eigenmode (done here after removing the interactions)
for i=1:size(midorder,1)
  if sum(midorder(i,:)>0)>1
   midtemp=midorder(i,[find(midorder(i,:)>0)]);
   SubAVG=AVG(midtemp,midtemp);
   for j=1:size(SubAVG,2)
      SubAVG(j,j)=0;
   end
   if max(eig(SubAVG))>Lmax
      Lmax=max(eig(SubAVG));
      imax=i;
   end
  end
end

midtemp=midorder(imax,[find(midorder(imax,:)>0)]);
SubAVG=AVG(midtemp,midtemp);

   for j=1:size(SubAVG,2)
      SubAVG(j,j)=0;
   end

[s,l]=eig(SubAVG);
Eig=max(diag(l))
[i,j]=find(l==max(diag(l)));
Mode(:,2)=s(:,i);
Mode(:,1)=midtemp';
for j=1:length(midtemp)
   Mode(j,1)=NAMES(Mode(j,1),1);
end
Mode
```

Simulation

dsmsimulate.m

```
function [WorkTimes,WorkPath,avg,sd]=dsmsimulate(DSM,LC,maxruns,timestep,order)
```

```
n=size(DSM,2);
%timestep should be less than or equal to min (duration/2 * impact) to capture work steps...

%sets up parameters for triangular distribution
%TaskDist=[diag(AVG)-0*ones(n,1),diag(AVG),diag(AVG)+0*ones(n,1)];

%get base sequencing order (may be passed in later)
%[frontorder,midorder,backorder,order]=dsmordernoswap(DSM(:,:,1));

%initialize learning curve
%LC=ones(n,1);

% reorder all matrices before starting runs
DSM=DSM([order],[order],:);
%TaskDist=TaskDist([order],:);
LC=LC([order],:);
DSMTemp=DSM;
run=0;

% loop starts here
while run < maxruns
   DSM=DSMTemp;
   run=run+1
   increment=1;
   time=0;

   %sample durations from triangular distribution if not reduced, lognormal if reduced
   for i=1:n
      if imag(DSM(i,i,4))==0
         DSM(i,i,1)=tripdf(DSM(i,i,1)-DSM(i,i,4),DSM(i,i,1),DSM(i,i,1)+DSM(i,i,4)); %here in
lieu of real task dist information!!!
      else
         DSM(i,i,1)=imag(DSM(i,i,4))+lognrnd(DSM(i,i,1),real(DSM(i,i,4))); %minimum plus
lognormal of difference
      end
      %TaskDist(i,4)=tripdf(TaskDist(i,1),TaskDist(i,2),TaskDist(i,3));
      for j=1:n
         if DSM(i,j,2)~=0
            DSM(i,j,2)=tripdf(max(DSM(i,j,2)-
DSM(i,j,4),0),DSM(i,j,2),min(1,DSM(i,j,2)+DSM(i,j,4))); %probabilities
            DSM(i,j,3)=tripdf(max(DSM(i,j,3)-
DSM(i,j,4),0),DSM(i,j,3),min(1,DSM(i,j,3)+DSM(i,j,4))); %impacts
         end
      end
   end
   TaskDist=diag(DSM(:,:,1));

   %initialize work variables and storage
   WorkRemaining=TaskDist;
   WorkedYet=zeros(n,1);
   WorkPath(1:n,increment,run)=WorkRemaining;
   WorkPath(n+1,increment,run)=time;

   %begin execution of run loop
   while sum(WorkRemaining>0)

      %sum(WorkRemaining>0)

      %figure out what can be worked now (if there's work to do and *upstream* precedents are
complete)
      WorkNow=zeros(n,1);
      for i=1:n
         WorkNow(i,1)=WorkRemaining(i,1)>0 & (sum(WorkRemaining(1:i-1,1)'.*DSM(i,1:i-
1,2))==0);
         WorkedYet(i,1)=WorkedYet(i,1)+WorkNow(i,1);
      end

      %process work in time step (just subtract the time)
      WorkRemaining=max(0,WorkRemaining-timestep*WorkNow);

      %if just finished in this round, add to dependency work (if probability satisfied, has been
worked already, include learning and impact effects), not sampling for added rework duration
      for i=1:n
         if WorkRemaining(i,1)==0 & WorkNow(i,1)==1

WorkRemaining=WorkRemaining+(rand(n,1)<DSM(:,i,2)).*(WorkedYet>0).*LC.*DSM(:,i,3).*
TaskDist;
         end
      end

      %advance and output results at new time
      time=time+timestep;
      increment=increment+1;
      WorkPath(1:n,increment,run)=WorkRemaining;
      WorkPath(n+1,increment,run)=time;
      WorkTimes(run)=time;

   end
   %Iterations(1:n,run)=WorkedYet./(TaskDist(:,4)/timestep);
end
```

```
hist(WorkTimes,100)
avg=mean(WorkTimes);
sd=std(WorkTimes);
```

Resource Model

dsmsimulateres.com

```
function
[ResourcePath,ResourceNeed,avg,sd]=dsmsimulateres(DSM,LC,ResourcesRequired,ResourcesA
vailable,maxruns,timestep)

n=size(DSM,2);
%ResourcesRequired=Resources(1:n,:);

%sets up parameters for triangular distribution
%TaskDist=[diag(AVG)-0*ones(n,1),diag(AVG),diag(AVG)+0*ones(n,1)];

%get base sequencing order (may be passed in later)
[frontorder,midorder,backorder,order]=dsmordernoswap(DSM(:,:,1));

%initialize learning curve
%LC=ones(n,1);

% reorder all matrices before starting runs
DSM=DSM([order],[order],:);
%TaskDist=TaskDist([order],:);
LC=LC([order],:);
ResourcesRequired=ResourcesRequired([order],:);
DSMTemp=DSM;

run=0;

% loop starts here
while run < maxruns
   DSM=DSMTemp;
   run=run+1
   increment=1;
   time=0;

   %sample durations from triangular distribution
   for i=1:n
      DSM(i,i,1)=tripdf(DSM(i,i,1)-DSM(i,i,4),DSM(i,i,1),DSM(i,i,1)+DSM(i,i,4));
      %TaskDist(i,4)=tripdf(TaskDist(i,1),TaskDist(i,2),TaskDist(i,3)); %durations
      for j=1:n
         if DSM(i,j,2)~=0
```

```
            DSM(i,j,2)=tripdf(max(DSM(i,j,2)-
DSM(i,j,4),0),DSM(i,j,2),min(1,DSM(i,j,2)+DSM(i,j,4))); %probabilities
            DSM(i,j,3)=tripdf(max(DSM(i,j,3)-
DSM(i,j,4),0),DSM(i,j,3),min(1,DSM(i,j,3)+DSM(i,j,4))); %impacts
         end
      end
   end

   TaskDist=diag(DSM(:,:,1));

   %initialize work variables and storage
   WorkRemaining=TaskDist;
   WorkedYet=zeros(n,1);
   WorkPath(1:n,increment,run)=WorkRemaining;
   WorkPath(n+1,increment,run)=time;
   %ResourcesAvailable=Resources(n+1,:);
   ResourcePath(increment,:,run)=ResourcesAvailable;
   WorkThen=zeros(n,1);

   %begin execution of run loop
   while sum(WorkRemaining>0)

      ResourceNeed(increment,:,run)=zeros(1,size(ResourcesAvailable,2));
      %figure out what can be worked now if there's work to do and ((*upstream* precedents are
complete and all resources are available) OR you worked on it the period before))
      WorkNow=zeros(n,1);
      for i=1:n
         WorkNow(i,1)=(WorkRemaining(i,1)>0) & (((sum(WorkRemaining(1:i-
1,1)'.*DSM(i,1:i-1,2))==0) & (prod((ResourcesAvailable-ResourcesRequired(i,:))>=0))) |
(WorkThen(i,1)==1));
         %record resource need, if applicable
         if prod((ResourcesAvailable-ResourcesRequired(i,:))>=0)==0 &
(WorkRemaining(i,1)>0) & (sum(WorkRemaining(1:i-1,1)'.*DSM(i,1:i-1,2))==0) &
(WorkThen(i,1)==0)

ResourceNeed(increment,:,run)=max(ResourceNeed(increment,:,run),((ResourcesAvailable-
ResourcesRequired(i,:))<0).*(ResourcesAvailable-ResourcesRequired(i,:))*-1); %treats each
max independently
         end
         WorkedYet(i,1)=WorkedYet(i,1)+WorkNow(i,1);
         %tie up resources in use immediately so they're not overallocated, also don't double
allocate
         if WorkNow(i,1)==1 & WorkThen(i,1)==0
            ResourcesAvailable=ResourcesAvailable-ResourcesRequired(i,:);
         end
      end
```

```
        %process work in time step (just subtract the time)
        WorkRemaining=max(0,WorkRemaining-timestep*WorkNow);

        %if just finished in this round, add to dependency work (if probability satisfied, has been
worked already, include learning and impact effects) and clear resources
        for i=1:n
            if WorkRemaining(i,1)==0 & WorkNow(i,1)==1

WorkRemaining=WorkRemaining+(rand(n,1)<DSM(:,i,2)).*(WorkedYet>0).*LC.*DSM(:,i,3).*
TaskDist;
            end
        end

        %free up resources if work remaining is still zero - must check after dependency loop is
closed to avoid adding back fake resources
        for i=1:n
            if WorkRemaining(i,1)==0 & WorkNow(i,1)==1
                ResourcesAvailable=ResourcesAvailable+ResourcesRequired(i,:);
            end
        end

        %advance and output results at new time
        time=time+timestep;
        increment=increment+1;
        WorkPath(1:n,increment,run)=WorkRemaining;
        WorkPath(n+1,increment,run)=time;
        ResourcePath(increment,:,run)=ResourcesAvailable;
        WorkTimes(run)=time;
        WorkThen=WorkNow;

    end
    %Iterations(1:n,run)=WorkedYet./(TaskDist(:,4)/timestep);
end

hist(WorkTimes,100)
avg=mean(WorkTimes);
sd=std(WorkTimes);
```

dsmsimulateresrelaxfactor.m

```
function
[ResourcePath,ResourcesNeed,ResourcesBudget,cost,avg,sd]=dsmsimulateresrelaxfactor(DSM,
LC,ResourcesRequired,ResourcesAvailable,maxruns,timestep,factor)
```

```
factor=max(0.5,min(factor,2)); %factor ranges between 0.5 and 2 only to restrict optimization to
realistic levels
n=size(DSM,2);
%ResourcesRequired=Resources(1:n,:);

%sets up parameters for triangular distribution
%TaskDist=[diag(AVG)-0*ones(n,1),diag(AVG),diag(AVG)+0*ones(n,1)];

%get base sequencing order (may be passed in later)
[frontorder,midorder,backorder,order]=dsmordernoswap(DSM(:,:,1));

%initialize learning curve
%LC=ones(n,1);

% reorder all matrices before starting runs
DSM=DSM([order],[order],:);
%TaskDist=TaskDist([order],:);
LC=LC([order],:);
ResourcesRequired=ResourcesRequired([order],:);
ResourcesBudget=ResourcesAvailable;
DSMTemp=DSM;
x=1;
costincrement=1;

%optimization loop
while x>0

clear ResourcesNeed;
run=0;

% runs loop starts here
while run < maxruns
   DSM=DSMTemp;
   run=run+1
   increment=1;
   time=0;

   %sample durations from triangular distribution
   for i=1:n
      if imag(DSM(i,i,4))==0
         DSM(i,i,1)=tripdf(DSM(i,i,1)-DSM(i,i,4),DSM(i,i,1),DSM(i,i,1)+DSM(i,i,4)); %here in
lieu of real task dist information!!!
      else
         DSM(i,i,1)=imag(DSM(i,i,4))+lognrnd(DSM(i,i,1),real(DSM(i,i,4))); %minimum plus
lognormal of difference
      end
```

```
        %TaskDist(i,4)=tripdf(TaskDist(i,1),TaskDist(i,2),TaskDist(i,3));
        %DSM(i,i,1)=tripdf(DSM(i,i,1)-DSM(i,i,4),DSM(i,i,1),DSM(i,i,1)+DSM(i,i,4));
        for j=1:n
            if DSM(i,j,2)~=0
                DSM(i,j,2)=tripdf(max(DSM(i,j,2)-
DSM(i,j,4),0),DSM(i,j,2),min(1,DSM(i,j,2)+DSM(i,j,4))); %probabilities
                DSM(i,j,3)=tripdf(max(DSM(i,j,3)-
DSM(i,j,4),0),DSM(i,j,3),min(1,DSM(i,j,3)+DSM(i,j,4))); %impacts
            end
        end
    end

    TaskDist=diag(DSM(:,:,1));

    %initialize work variables and storage
    WorkRemaining=TaskDist;
    WorkedYet=zeros(n,1);
    WorkPath(1:n,increment,run)=WorkRemaining;
    WorkPath(n+1,increment,run)=time;
    ResourcesAvailable=ResourcesBudget;
    ResourcePath(increment,:,run)=ResourcesAvailable;
    WorkThen=zeros(n,1);

    %begin execution of run loop
    while sum(WorkRemaining>0)

        %sum(WorkRemaining>0)

        ResourcesNeed(increment,:,run)=zeros(1,size(ResourcesAvailable,2));
        %figure out what can be worked now if there's work to do and ((*upstream* precedents are
complete and all resources are available) OR you worked on it the period before))
        WorkNow=zeros(n,1);
        for i=1:n
            WorkNow(i,1)=(WorkRemaining(i,1)>0) & (((sum(WorkRemaining(1:i-
1,1)'.*DSM(i,1:i-1,2))==0) & (prod((ResourcesAvailable-factor*ResourcesRequired(i,:))>=0))) |
(WorkThen(i,1)==1));
            %record resource need, if applicable
            if prod((ResourcesAvailable-factor*ResourcesRequired(i,:))>=0)==0 &
(WorkRemaining(i,1)>0) & (sum(WorkRemaining(1:i-1,1)'.*DSM(i,1:i-1,2))==0) &
(WorkThen(i,1)==0)

ResourcesNeed(increment,:,run)=max(ResourcesNeed(increment,:,run),((ResourcesAvailable-
factor*ResourcesRequired(i,:))<0).*(ResourcesAvailable-factor*ResourcesRequired(i,:))*-1);
%treats each max independently
            end
            WorkedYet(i,1)=WorkedYet(i,1)+WorkNow(i,1);
        %tie up resources in use immediately so they're not overallocated, also don't double
allocate
        if WorkNow(i,1)==1 & WorkThen(i,1)==0
            ResourcesAvailable=ResourcesAvailable-factor*ResourcesRequired(i,:);
        end
    end

    %process work in time step (just subtract the time)
    WorkRemaining=max(0,WorkRemaining-factor*timestep*WorkNow);

    %if just finished in this round, add to dependency work (if probability satisfied, has been
worked already, include learning and impact effects) and clear resources
    for i=1:n
        if WorkRemaining(i,1)==0 & WorkNow(i,1)==1

WorkRemaining=WorkRemaining+(rand(n,1)<DSM(:,i,2)).*(WorkedYet>0).*LC.*DSM(:,i,3).*
TaskDist;
        end
    end

    %free up resources if work remaining is still zero - must check after dependency loop is
closed to avoid adding back fake resources
    for i=1:n
        if WorkRemaining(i,1)==0 & WorkNow(i,1)==1
            ResourcesAvailable=ResourcesAvailable+factor*ResourcesRequired(i,:);
        end
    end

    %advance and output results at new time
    time=time+timestep;
    increment=increment+1;
    WorkPath(1:n,increment,run)=WorkRemaining;
    WorkPath(n+1,increment,run)=time;
    ResourcePath(increment,:,run)=ResourcesAvailable;
    WorkTimes(run)=time;
    WorkThen=WorkNow;

  end
  %Iterations(1:n,run)=WorkedYet./(TaskDist(:,4)/timestep);
end

%hist(WorkTimes,100)
avg=mean(WorkTimes);
sd=std(WorkTimes);

cost(costincrement,1)=x;
cost(costincrement,2)=sum(ResourcesBudget)*mean(WorkTimes);
cost(costincrement,3)=mean(WorkTimes);
costincrement=costincrement+1;

%Add needed resources to budget
ResourcesBudget=ResourcesBudget+ceil(mean(mean(ResourcesNeed,3)));
x=sum(mean(mean(ResourcesNeed,3)));

end
cost
plot(cost(:,2),cost(:,1),'b+')
```

Multiple Projects

dsmsimulateresmult.m

```
function
[ResourcePath,ResourceNeed,WorkPath,WorkRemaining,WorkTimes,avg,sd]=dsmsimulateresm
ult(DSM,LC,ResourcesRequired,ResourcesAvailable,delays,maxruns,timestep)

%one resource pool...

projects=size(DSM,2)
%initialize WorkTimes
WorkTimes=cell(1,projects);

%prepare each project...
for project=1:projects

   %get number of tasks and pull resource requirements
   n(project)=size(DSM{project},2);
   %ResourcesRequired{project}=Resources{project}(1:n(project),:);

   %get base sequencing order (may be passed in later)
   [frontorder,midorder,backorder,order{project}]=dsmordernoswap(DSM{project}(:,:,1));

   % reorder all matrices before starting runs
   DSM{project}=DSM{project}([order{project}],[order{project}],:);
   %TaskDist{project}=TaskDist{project}([order{project}],:);
   LC{project}=LC{project}([order{project}],:);
   ResourcesRequired{project}=ResourcesRequired{project}([order{project}],:);
   DSMTemp{project}=DSM{project};

end
```

```
run=0;

% loop starts here
while run < maxruns

   run=run+1
   increment=1;
   time=0;

   %reset each project's data
   for project=1:projects

      DSM{project}=DSMTemp{project};

      %sample durations from triangular distribution
      for i=1:n(project)
         if imag(DSM{project}(i,i,4))==0
            DSM{project}(i,i,1)=tripdf(DSM{project}(i,i,1)-
DSM{project}(i,i,4),DSM{project}(i,i,1),DSM{project}(i,i,1)+DSM{project}(i,i,4)); %here in
lieu of real task dist information!!!
         else

DSM{project}(i,i,1)=imag(DSM{project}(i,i,4))+lognrnd(DSM{project}(i,i,1),real(DSM{projec
t}(i,i,4))); %minimum plus lognormal of difference
         end


%TaskDist{project}(i,4)=tripdf(TaskDist{project}(i,1),TaskDist{project}(i,2),TaskDist{project}
(i,3));

         for j=1:n
            if DSM{project}(i,j,2)~=0
               DSM{project}(i,j,2)=tripdf(max(DSM{project}(i,j,2)-
DSM{project}(i,j,4),0),DSM{project}(i,j,2),min(1,DSM{project}(i,j,2)+DSM{project}(i,j,4)));
%probabilities
               DSM{project}(i,j,3)=tripdf(max(DSM{project}(i,j,3)-
DSM{project}(i,j,4),0),DSM{project}(i,j,3),min(1,DSM{project}(i,j,3)+DSM{project}(i,j,4)));
%impacts
            end
         end
      end

      TaskDist{project}=diag(DSM{project}(:,:,1));

      %initialize work variables and storage
      WorkRemaining{project}=TaskDist{project};
```

```
        WorkedYet{project}=zeros(n(project),1);
        WorkPath{project}(1:n(project),increment,run)=WorkRemaining{project};
        WorkPath{project}(n(project)+1,increment,run)=time;
        %ResourcesAvailable{project}=Resources{project}(n(project)+1,:);
        ResourcePath(increment,:,run)=ResourcesAvailable;
        WorkThen{project}=zeros(n(project),1);
        WorkNow{project}=zeros(n(project),1);

    end

    %begin execution of run loop
    while sum(sum(cell2mat(WorkRemaining)))>0

        sum(sum(cell2mat(WorkRemaining)))

        for project=1:projects %must run all projects
        if time>=delays(project) %but only run them after their required delay

        ResourceNeed(increment,:,run)=zeros(1,size(ResourcesAvailable,2));
        %figure out what can be worked now if there's work to do and ((*upstream* precedents are
complete and all resources are available) OR you worked on it the period before))
        WorkNow{project}=zeros(n(project),1);
        for i=1:n(project)
            WorkNow{project}(i,1)=(WorkRemaining{project}(i,1)>0) &
(((sum(WorkRemaining{project}(1:i-1,1)'.*DSM{project}(i,1:i-1,2))==0) &
(prod((ResourcesAvailable-ResourcesRequired{project}(i,:))>=0))) |
(WorkThen{project}(i,1)==1));
            %record resource need, if applicable
            if prod((ResourcesAvailable-ResourcesRequired{project}(i,:))>=0)==0 &
(WorkRemaining{project}(i,1)>0) & (sum(WorkRemaining{project}(1:i-
1,1)'.*DSM{project}(i,1:i-1,2))==0) & (WorkThen{project}(i,1)==0)

ResourceNeed(increment,:,run)=max(ResourceNeed(increment,:,run),((ResourcesAvailable-
ResourcesRequired{project}(i,:))<0).*(ResourcesAvailable-ResourcesRequired{project}(i,:))*-
1); %treats each max independently
            end
            WorkedYet{project}(i,1)=WorkedYet{project}(i,1)+WorkNow{project}(i,1);
            %tie up resources in use immediately so they're not overallocated, also don't double
allocate
            if WorkNow{project}(i,1)==1 & WorkThen{project}(i,1)==0
                ResourcesAvailable=ResourcesAvailable-ResourcesRequired{project}(i,:);
            end
        end

        %process work in time step (just subtract the time)
        WorkRemaining{project}=max(0,WorkRemaining{project}-timestep*WorkNow{project});
        %if just finished in this round, add to dependency work (if probability satisfied, has been
worked already, include learning and impact effects) and clear resources
        for i=1:n(project)
            if WorkRemaining{project}(i,1)==0 & WorkNow{project}(i,1)==1

WorkRemaining{project}=WorkRemaining{project}+(rand(n(project),1)<DSM{project}(:,i,2)).
*(WorkedYet{project}>0).*LC{project}.*DSM{project}(:,i,3).*TaskDist{project};
            end
        end

        %free up resources if work remaining is still zero - must check after dependency loop is
closed to avoid adding back fake resources
        for i=1:n(project)
            if WorkRemaining{project}(i,1)==0 & WorkNow{project}(i,1)==1
                ResourcesAvailable=ResourcesAvailable+ResourcesRequired{project}(i,:);
            end
        end

        end %end if statement
        end %end project loop

        %advance time and increment
        time=time+timestep;
        increment=increment+1;

        %output results at new time
        for project=1:projects
            WorkPath{project}(1:n(project),increment,run)=WorkRemaining{project};
            WorkPath{project}(n(project)+1,increment,run)=time;
            ResourcePath(increment,:,run)=ResourcesAvailable;
            if sum(WorkPath{project}(1:n(project),increment,run))==0 &
sum(WorkPath{project}(1:n(project),increment-1,run))>0
                WorkTimes{project}(run)=time;
            end
            WorkThen{project}=WorkNow{project};
        end

    end %end execution loop
end %end run loop

Work=0;

for project=1:projects
    %hist(WorkTimes{project},100)
    Work=Work+sum(mean(WorkPath{project}(1:n(project),:,:),3));
```

```
   avg(project)=mean(WorkTimes{project});
   sd(project)=std(WorkTimes{project});
end

%Work(2,:)=[0:timestep:max(WorkTimes{projects})];
%plot(Work(2,:),Work(1,:)/max(Work(1,:)))
```

dsmsimulateresmultopt.m

```
function
[ResourcePath,ResourceNeed,WorkPath,WorkRemaining,WorkTimes,DelayTimes,avg,sd]=dsm
simulateresmultopt(DSM,LC,ResourcesRequired,ResourcesAvailable,maxruns,timestep)

%in principle, easily extended to 3 or more projects - just map delays in order...
%also written to do non-indentical projects

%prepare each project...
projects=size(DSM,2);

for project=1:projects

   %get number of tasks and pull resource requirements
   n(project)=size(DSM{project},2);
   %ResourcesRequired{project}=Resources{project}(1:n(project),:);

   %get base sequencing order (may be passed in later)
   [frontorder,midorder,backorder,order{project}]=dsmordernoswap(DSM{project}(:,:,1));

   % reorder all matrices before starting runs
   DSM{project}=DSM{project}([order{project}],[order{project}],:);
   %TaskDist{project}=TaskDist{project}([order{project}],:);
   LC{project}=LC{project}([order{project}],:);
   ResourcesRequired{project}=ResourcesRequired{project}([order{project}],:);
   DSMTemp{project}=DSM{project};

end

delayincrement=1;

for delay=0:10:100 %make sure timestep is less than delay increment!

delays=[0 delay]

%initialize WorkTimes and clear variables that have changing size (anything with "increment" in
the index)
clear WorkPath;
clear ResourcePath;
WorkTimes=cell(1,projects);

run=0;

% loop starts here
while run < maxruns

   run=run+1
   increment=1;
   time=0;

   %reset each project's data
   for project=1:projects

      DSM{project}=DSMTemp{project};

      %sample durations from triangular distribution
      for i=1:n(project)
         if imag(DSM{project}(i,i,4))==0
            DSM{project}(i,i,1)=tripdf(DSM{project}(i,i,1)-
DSM{project}(i,i,4),DSM{project}(i,i,1),DSM{project}(i,i,1)+DSM{project}(i,i,4)); %here in
lieu of real task dist information!!!
         else

DSM{project}(i,i,1)=imag(DSM{project}(i,i,4))+lognrnd(DSM{project}(i,i,1),real(DSM{projec
t}(i,i,4))); %minimum plus lognormal of difference
         end


%TaskDist{project}(i,4)=tripdf(TaskDist{project}(i,1),TaskDist{project}(i,2),TaskDist{project}
(i,3));
         %DSM{project}(i,i,1)=tripdf(DSM{project}(i,i,1)-
DSM{project}(i,i,4),DSM{project}(i,i,1),DSM{project}(i,i,1)+DSM{project}(i,i,4));

         for j=1:n
            if DSM{project}(i,j,2)~=0
               DSM{project}(i,j,2)=tripdf(max(DSM{project}(i,j,2)-
DSM{project}(i,j,4),0),DSM{project}(i,j,2),min(1,DSM{project}(i,j,2)+DSM{project}(i,j,4)));
%probabilities
               DSM{project}(i,j,3)=tripdf(max(DSM{project}(i,j,3)-
DSM{project}(i,j,4),0),DSM{project}(i,j,3),min(1,DSM{project}(i,j,3)+DSM{project}(i,j,4)));
%impacts
            end
         end
      end
```

```
        TaskDist{project}=diag(DSM{project}(:,:,1));

        %initialize work variables and storage
        WorkRemaining{project}=TaskDist{project};
        WorkedYet{project}=zeros(n(project),1);
        WorkPath{project}(1:n(project),increment,run)=WorkRemaining{project};
        WorkPath{project}(n(project)+1,increment,run)=time;
        %ResourcesAvailable{project}=Resources{project}(n(project)+1,:);
        ResourcePath(increment,:,run)=ResourcesAvailable;
        WorkThen{project}=zeros(n(project),1);
        WorkNow{project}=zeros(n(project),1);

    end

    %begin execution of run loop
    while sum(sum(cell2mat(WorkRemaining)))>0

        sum(sum(cell2mat(WorkRemaining)))

        for project=1:projects %must run all projects
        if time>=delays(project) %but only run them after their required delay

        ResourceNeed{project}(increment,:,run)=zeros(1,size(ResourcesAvailable,2));
        %figure out what can be worked now if there's work to do and ((*upstream* precedents are
complete and all resources are available) OR you worked on it the period before))
        WorkNow{project}=zeros(n(project),1);
        for i=1:n(project)
            WorkNow{project}(i,1)=(WorkRemaining{project}(i,1)>0) &
(((sum(WorkRemaining{project}(1:i-1,1)'.*DSM{project}(i,1:i-1,2))==0) &
(prod((ResourcesAvailable-ResourcesRequired{project}(i,:))>=0))) |
(WorkThen{project}(i,1)==1));
            %record resource need, if applicable
            if prod((ResourcesAvailable-ResourcesRequired{project}(i,:))>=0)==0 &
(WorkRemaining{project}(i,1)>0) & (sum(WorkRemaining{project}(1:i-
1,1)'.*DSM{project}(i,1:i-1,2))==0) & (WorkThen{project}(i,1)==0)

ResourceNeed{project}(increment,:,run)=max(ResourceNeed{project}(increment,:,run),((Resou
rcesAvailable-ResourcesRequired{project}(i,:))<0).*(ResourcesAvailable-
ResourcesRequired{project}(i,:))*-1); %treats each max independently
            end
            WorkedYet{project}(i,1)=WorkedYet{project}(i,1)+WorkNow{project}(i,1);
            %tie up resources in use immediately so they're not overallocated, also don't double
allocate
            if WorkNow{project}(i,1)==1 & WorkThen{project}(i,1)==0
                ResourcesAvailable=ResourcesAvailable-ResourcesRequired{project}(i,:);
            end
        end

        %process work in time step (just subtract the time)
        WorkRemaining{project}=max(0,WorkRemaining{project}-timestep*WorkNow{project});

        %if just finished in this round, add to dependency work (if probability satisfied, has been
worked already, include learning and impact effects) and clear resources
        for i=1:n(project)
            if WorkRemaining{project}(i,1)==0 & WorkNow{project}(i,1)==1

WorkRemaining{project}=WorkRemaining{project}+(rand(n(project),1)<DSM{project}(:,i,2)).
*(WorkedYet{project}>0).*LC{project}.*DSM{project}(:,i,3).*TaskDist{project};
            end
        end

        %free up resources if work remaining is still zero - must check after dependency loop is
closed to avoid adding back fake resources
        for i=1:n(project)
            if WorkRemaining{project}(i,1)==0 & WorkNow{project}(i,1)==1
                ResourcesAvailable=ResourcesAvailable+ResourcesRequired{project}(i,:);
            end
        end

        end %end if statement
        end %end project loop

        %advance time and increment
        time=time+timestep;
        increment=increment+1;

        %output results at new time
        for project=1:projects
            WorkPath{project}(1:n(project),increment,run)=WorkRemaining{project};
            WorkPath{project}(n(project)+1,increment,run)=time;
            ResourcePath(increment,:,run)=ResourcesAvailable;
            if sum(WorkPath{project}(1:n(project),increment,run))==0 &
sum(WorkPath{project}(1:n(project),increment-1,run))>0
                WorkTimes{project}(run)=time;
            end
            WorkThen{project}=WorkNow{project};
        end

    end %end execution loop
end %end run loop
```

```
Work=0;
CumWorkTimes(delayincrement,1)=delay;

for project=1:projects
   %hist(WorkTimes{project},100)
   Work=Work+sum(mean(WorkPath{project}(1:n(project),:,:),3));
   avg(project)=mean(WorkTimes{project});
   sd(project)=std(WorkTimes{project});
   CumWorkTimes(delayincrement,project+1)=avg(project)-delays(project);
end

%average work profile
%Work(2,:)=[0:timestep:max(WorkTimes{projects})];
%plot(Work(2,:),Work(1,:)/max(Work(1,:)))

DelayTimes(delayincrement,1)=delay;
DelayTimes(delayincrement,2)=max(avg);
DelayTimes(delayincrement,3)=sd(find(avg==max(avg)));
CumWorkTimes(delayincrement,projects+2)=sum(CumWorkTimes(delayincrement,2:projects+
1));

delayincrement=delayincrement+1;

end

plot(DelayTimes(:,1),DelayTimes(:,2)) %time to finish
%figure
%plot(CumWorkTimes(:,1),CumWorkTimes(:,end)) %cumulative work done (days)
%figure
%plot(CumWorkTimes(:,1),CumWorkTimes(:,end)./DelayTimes(:,2)) %utilization
```

dsmsimulateresmultrelaxfactor.m

```
function
[ResourcePath,ResourcesNeed,ResourcesBudget,WorkPath,WorkRemaining,WorkTimes,cost,av
g,sd]=dsmsimulateresmultrelaxfactor(DSM,LC,ResourcesRequired,ResourcesAvailable,delays,
maxruns,timestep,factor)

%one resource pool...
factor=max(0.5,min(factor,2)); %factor ranges between 0.5 and 2 only to restrict optimization to
realistic levels

projects=size(DSM,2)
%initialize WorkTimes
WorkTimes=cell(1,projects);
```

```
%prepare each project...
for project=1:projects

   %get number of tasks and pull resource requirements
   n(project)=size(DSM{project},2);
   %ResourcesRequired{project}=Resources{project}(1:n(project),:);

   %get base sequencing order (may be passed in later)
   [frontorder,midorder,backorder,order{project}]=dsmordernoswap(DSM{project}(:,:,1));

   % reorder all matrices before starting runs
   DSM{project}=DSM{project}([order{project}],[order{project}],:);
   %TaskDist{project}=TaskDist{project}([order{project}],:);
   LC{project}=LC{project}([order{project}],:);
   ResourcesRequired{project}=ResourcesRequired{project}([order{project}],:);
   DSMTemp{project}=DSM{project};

end

ResourcesBudget=ResourcesAvailable;
x=1;
costincrement=1;

%budget optimization loop
while x>0

%initialize WorkTimes and clear variables that have changing size (anything with "increment" in
the index)
clear WorkPath;
clear ResourcePath;
WorkTimes=cell(1,projects);
clear ResourcesNeed;
run=0;

% loop starts here
while run < maxruns

   run=run+1
   increment=1;
   time=0;

   %reset each project's data
   for project=1:projects

      DSM{project}=DSMTemp{project};
```

```
        %sample durations from triangular distribution
        for i=1:n(project)
           if imag(DSM{project}(i,i,4))==0
              DSM{project}(i,i,1)=tripdf(DSM{project}(i,i,1)-
DSM{project}(i,i,4),DSM{project}(i,i,1),DSM{project}(i,i,1)+DSM{project}(i,i,4)); %here in
lieu of real task dist information!!!
           else

DSM{project}(i,i,1)=imag(DSM{project}(i,i,4))+lognrnd(DSM{project}(i,i,1),real(DSM{projec
t}(i,i,4))); %minimum plus lognormal of difference
           end

%TaskDist{project}(i,4)=tripdf(TaskDist{project}(i,1),TaskDist{project}(i,2),TaskDist{project}
(i,3));
           %DSM{project}(i,i,1)=tripdf(DSM{project}(i,i,1)-
DSM{project}(i,i,4),DSM{project}(i,i,1),DSM{project}(i,i,1)+DSM{project}(i,i,4));
           for j=1:n
              if DSM{project}(i,j,2)~=0
                 DSM{project}(i,j,2)=tripdf(max(DSM{project}(i,j,2)-
DSM{project}(i,j,4),0),DSM{project}(i,j,2),min(1,DSM{project}(i,j,2)+DSM{project}(i,j,4)));
%probabilities
                 DSM{project}(i,j,3)=tripdf(max(DSM{project}(i,j,3)-
DSM{project}(i,j,4),0),DSM{project}(i,j,3),min(1,DSM{project}(i,j,3)+DSM{project}(i,j,4)));
%impacts
              end
           end
        end

        TaskDist{project}=diag(DSM{project}(:,:,1));

        %initialize work variables and storage
        WorkRemaining{project}=TaskDist{project};
        WorkedYet{project}=zeros(n(project),1);
        WorkPath{project}(1:n(project),increment,run)=WorkRemaining{project};
        WorkPath{project}(n(project)+1,increment,run)=time;
        ResourcesAvailable=ResourcesBudget;
        ResourcePath(increment,:,run)=ResourcesAvailable;
        WorkThen{project}=zeros(n(project),1);
        WorkNow{project}=zeros(n(project),1);

     end

     %begin execution of run loop
     while sum(sum(cell2mat(WorkRemaining)))>0

        %sum(sum(cell2mat(WorkRemaining)))
```

```
    for project=1:projects %must run all projects
    if time>=delays(project) %but only run them after their required delay

    ResourcesNeed(increment,:,run)=zeros(1,size(ResourcesAvailable,2));
    %figure out what can be worked now if there's work to do and ((*upstream* precedents are
complete and all resources are available) OR you worked on it the period before))
    WorkNow{project}=zeros(n(project),1);
    for i=1:n(project)
      WorkNow{project}(i,1)=(WorkRemaining{project}(i,1)>0) &
(((sum(WorkRemaining{project}(1:i-1,1)'.*DSM{project}(i,1:i-1,2))==0) &
(prod((ResourcesAvailable-factor*ResourcesRequired{project}(i,:))>=0))) |
(WorkThen{project}(i,1)==1));
      %record resource need, if applicable
      if prod((ResourcesAvailable-factor*ResourcesRequired{project}(i,:))>=0)==0 &
(WorkRemaining{project}(i,1)>0) & (sum(WorkRemaining{project}(1:i-
1,1)'.*DSM{project}(i,1:i-1,2))==0) & (WorkThen{project}(i,1)==0)

ResourcesNeed(increment,:,run)=max(ResourcesNeed(increment,:,run),((ResourcesAvailable-
factor*ResourcesRequired{project}(i,:))<0).*(ResourcesAvailable-
factor*ResourcesRequired{project}(i,:))*-1); %treats each max independently
      end
      WorkedYet{project}(i,1)=WorkedYet{project}(i,1)+WorkNow{project}(i,1);
      %tie up resources in use immediately so they're not overallocated, also don't double
allocate
      if WorkNow{project}(i,1)==1 & WorkThen{project}(i,1)==0
        ResourcesAvailable=ResourcesAvailable-factor*ResourcesRequired{project}(i,:);
      end
    end

    %process work in time step (just subtract the time)
    WorkRemaining{project}=max(0,WorkRemaining{project}-
factor*timestep*WorkNow{project});

    %if just finished in this round, add to dependency work (if probability satisfied, has been
worked already, include learning and impact effects) and clear resources
    for i=1:n(project)
      if WorkRemaining{project}(i,1)==0 & WorkNow{project}(i,1)==1

WorkRemaining{project}=WorkRemaining{project}+(rand(n(project),1)<DSM{project}(:,i,2)).
*(WorkedYet{project}>0).*LC{project}.*DSM{project}(:,i,3).*TaskDist{project};
      end
    end

    %free up resources if work remaining is still zero - must check after dependency loop is
closed to avoid adding back fake resources
```

```
      for i=1:n(project)
         if WorkRemaining{project}(i,1)==0 & WorkNow{project}(i,1)==1
            ResourcesAvailable=ResourcesAvailable+factor*ResourcesRequired{project}(i,:);
         end
      end

      end %end if statement
      end %end project loop

      %advance time and increment
      time=time+timestep;
      increment=increment+1;

      %output results at new time
      for project=1:projects
         WorkPath{project}(1:n(project),increment,run)=WorkRemaining{project};
         WorkPath{project}(n(project)+1,increment,run)=time;
         ResourcePath(increment,:,run)=ResourcesAvailable;
         if sum(WorkPath{project}(1:n(project),increment,run))==0 &
sum(WorkPath{project}(1:n(project),increment-1,run))>0
            WorkTimes{project}(run)=time;
         end
         WorkThen{project}=WorkNow{project};
      end

   end %end execution loop
end %end run loop

Work=0;

for project=1:projects
   %hist(WorkTimes{project},100)
   Work=Work+sum(mean(WorkPath{project}(1:n(project),:,:),3));
   avg(project)=mean(WorkTimes{project});
   sd(project)=std(WorkTimes{project});
end

%Work(2,:)=[0:timestep:max(WorkTimes{projects})];
%plot(Work(2,:),Work(1,:)/max(Work(1,:)));

cost(costincrement,1)=x;
cost(costincrement,2)=sum(ResourcesBudget)*max(avg);
cost(costincrement,3)=max(avg);
costincrement=costincrement+1;

%Add needed resources to budget
```

```
ResourcesBudget=ResourcesBudget+ceil(mean(mean(ResourcesNeed,3)));
x=sum(mean(mean(ResourcesNeed,3)));

end %end budget loop
%cost
%plot(cost(:,3),cost(:,2),'b+')
```