

Methodology for Architecture Development for Product Design

by

Michael J. Martin

M.S. Mechanical Engineering  
University of Rochester, 1987

Submitted to the System Design and Management Program  
in Partial Fulfillment of the Requirements for the  
Degree of Masters of Science in Engineering and Management

at the

Massachusetts Institute of Technology

February 2000

© 1999 Michael John Martin. All rights reserved.

The author hereby grants MIT permission to reproduce and distribute publicly  
paper and electronic copies of this document in whole or in part

Signature of author

*M. J. Martin*

System Design and Management Program  
December 13, 1999

Certified by

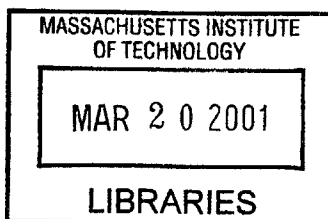
*Daniel Frey*  
Daniel Frey  
Assistant Professor Aeronautics & Astronautics  
December 13, 1999

Accepted by

*Thomas A. Kochan*  
Thomas A. Kochan  
LFM/SDM Co-Director  
George M. Bunker Professor of Management

Accepted by

*Paul A. Lagace*  
Paul A. Lagace  
LFM/SDM Co-Director  
Professor of Aeronautics & Astronautics and  
Engineering Systems



ENG



# Methodology for Architecture Development for Product Design

by

Michael J. Martin

Submitted to the System Design and Management Program  
on December 13, 1999 in Partial Fulfillment of the Requirements for the  
Degree of Masters of Science in Engineering and Management

## **ABSTRACT**

An integrative methodology for architecture development in a product development environment is described. The methodology combines the use of the design structure matrix technique with constraint-based modeling to create a process that satisfies the following requirements:

1. Provide a means for modeling the system that provides the capability to gain feedback on proposed decisions. This promotes rapidly system learning.
2. Provide a definition of the linkage between product requirements and design parameters.
3. Provide documentation that makes the architecture explicit and enables others to have access to the architectural knowledge.
4. Increase confidence in the proposed system so that product design can proceed with a minimum of risk.

The application of the methodology in the context of the development of the xerographic module architecture for color printing system is described. The project was a clean sheet design using a new color architecture and implementing seven new technologies. A significant result is that once the architecture was accepted and placed under change control, the architecture has not changed in four years. Traditionally, similar projects have had to make significant changes as the design matured.

Based on the case study, there is anecdotal evidence to support the hypothesis that the methodology can be successfully used to develop complex systems. It is shown that the methodology is closely aligned to the product development process. During the pre-concept and concept phases, the models were used to develop the system architecture. During the detailed design phase, the models can be used to maintain the integrity of the architecture as the design and technologies mature. Finally, in order for the methodology to be successfully applied it must have the full support of program management and the design and technology organizations.

Thesis Supervisor: Daniel Frey  
Title: Assistant Professor Aeronautics & Astronautics



## ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Dan Frey, for this help and support in developing this publication. His efforts have significantly improved the quality of this work.

I would also like to thank Ralph Gifford and Dave Cronin from Cognition Corp. for the excellent technical support and the loan of a copy of the Mechanical Advantage application.

I am indebted to the many members of the Xerox Print Engine Development Unit who have supported and encouraged my efforts over the last two years. In particular I would like to thank the following people: Ken Blanchard for nominating me for the SDM program, without his continuous support and mentoring I would not be writing this, Jim Franzen for suggesting the initial work on which this thesis is based and for the many years of friendship and mentoring, Mark Enzien for understanding the constraints that the SDM program places on the student-worker, and for helping to achieve a balance between work, school, and family demands, and Joe Wing for helping to refine the elements of the methodology.

I would like to thank my classmates in the SDM98 class. I am proud to be a part of this truly fantastic group of people. I am especially grateful to my Rochester cohorts Shelley, Rob, Mike, Joe, Tom, and Karl, whose intelligence, work ethic, and sense of humor made the past two years an unforgettable experience.

I am forever thankful to my parents for being there whenever I needed them and for instilling in me the desire to learn.

To my children, Mary Emma and Molly, for providing moments of pure joy and much needed relief from long hours of studying. I have never worked harder than I have in the past two years and I have never laughed or smiled more, most of the smiles were from you.

This thesis is dedicated to my wife and best friend, Katy. Without her commitment and support, I could not have completed this program. The past two years have required tremendous patience and understanding and I am forever thankful.



## TABLE OF CONTENTS

<b>1</b>	<b>PROBLEM REVIEW .....</b>	<b>9</b>
<b>2</b>	<b>REVIEW OF PRIOR WORK – TOWARDS A FRAMEWORK.....</b>	<b>11</b>
2.1	DESIGN STRUCTURE MATRIX TECHNIQUE .....	18
2.2	CONSTRAINT SATISFACTION PROBLEMS .....	27
<b>3</b>	<b>IMPLEMENTATION OF PRACTICAL METHODOLOGY .....</b>	<b>29</b>
3.1	PROCESS CONTEXT .....	30
3.2	ORGANIZATIONAL CONTEXT.....	31
3.3	METHODOLOGY DESCRIPTION.....	32
3.4	TOOLS TO SUPPORT METHODOLOGY .....	35
3.5	ROLE OF THE ARCHITECT .....	39
3.6	METHODOLOGY FLOW DIAGRAM .....	45
<b>4</b>	<b>CASE STUDY -- XEROGRAPHIC COLOR ARCHITECTURE .....</b>	<b>46</b>
4.1	CASE INTRODUCTION .....	46
4.2	ARCHITECTURAL PARTITIONING.....	46
4.3	XEROGRAPHIC MODULE .....	47
4.4	GROWTH IN COMPLEXITY .....	51
4.5	CASE STUDY .....	53
4.6	PRODUCT REQUIREMENTS, DESIGNS RULES, AND GEOMETRIC CONSTRAINTS .	57
4.7	A CLARIFYING EXAMPLE.....	59
4.8	XEROGRAPHIC ARCHITECTURE MODEL.....	66
4.9	ORGANIZATIONAL STRUCTURE AND ROLES AND RESPONSIBILITIES .....	76
4.10	ALIGNMENT OF METHODOLOGY WITH PDP PROCESS .....	77
4.11	PROJECT TIMELINE .....	83
4.12	LESSON'S LEARNED .....	86
4.13	UNIQUE VALUE CREATED BY SYSTEM/ARCHITECTURE MODEL .....	87
<b>5</b>	<b>CONCLUSIONS AND SUMMARY.....</b>	<b>88</b>
<b>6</b>	<b>APPENDIX A: PRINTING MACHINE ARCHITECTURE .....</b>	<b>96</b>
<b>7</b>	<b>REFERENCES.....</b>	<b>101</b>

LIST OF FIGURES

FIGURE 1: STERMAN'S MODEL FOR LEARNING ..... 14

FIGURE 2: SAMPLE DSM ..... 18

FIGURE 3: PARTITIONED DSM ..... 20

FIGURE 4: CLUSTERED DSM ..... 21

FIGURE 5: DSM EXAMPLE FOR PRINT SPEED ..... 26

FIGURE 6: PRODUCT DELIVERY PROCESS ..... 30

FIGURE 7: INFLUENCES ON SYSTEM ARCHITECTURE ..... 31

FIGURE 8: PROCESS FLOW FOR INTEGRATIVE ARCHITECTURE DEVELOPMENT  
METHODOLOGY ..... 33

FIGURE 9: EXAMPLE IN MECHANICAL ADVANTAGE ..... 38

FIGURE 10: PHYSICAL DOMAIN AND MODEL BOUNDARIES ..... 41

FIGURE 11: METHODOLOGY FLOW DIAGRAM ..... 45

FIGURE 12: MONOCHROME XEROGRAPHIC ENGINE ..... 48

FIGURE 13: SINGLE-PASS MULTI-COLOR ARCHITECTURE ..... 51

FIGURE 14: COMPARISON OF ARCHITECTURES ..... 52

FIGURE 15: DEFINITION OF BOUNDARY ..... 55

FIGURE 16: MAPPING OF MODEL SPACE ..... 57

FIGURE 17: IMAGE PITCH LENGTH DIAGRAM ..... 60

FIGURE 18: IMAGE PITCH LENGTH DSM ..... 62

FIGURE 19: IMAGE PITCH RELEVANT DIMENSIONS ..... 63

FIGURE 20: ARCHITECTURE DSM ..... 70

FIGURE 21: SHOW CONNECTED ITEMS ..... 71

FIGURE 22: WATERFRONT ILLUSTRATION ..... 73

FIGURE 23: PROJECT TIMELINE ..... 85

FIGURE 24: METHODOLOGY MAPPED TO THE MODEL OF LEARNING ..... 94



## **1 PROBLEM REVIEW**

Developing a new system is a significant undertaking requiring the expenditure of numerous resources to ensure that the system meets the intended needs. The task becomes more formidable when the system is based on a new architecture. If new subsystem and component technologies are involved then the task of developing the system is even more complex as the knowledge of interactions is less certain. Managing the product development of a complex system is prone to delays due to lack of knowledge of system interactions, poor design decisions, and unanticipated performance shortfalls. These problems usually cause a delay in the bringing the product to market. If the intended market is rapidly changing, then being late to market can mean that competitors have gained the first mover's advantage and have gained a strong position in the market. Given enough of an advantage, the competition might be able to achieve system lock-in and command a majority share of the market profits.

As an illustration of the complexity in existing products, Eppinger [8] has reported that a copier redesign requires 400 people, 125 subassemblies, 2000 engineering drawings and over 1 million decisions. Many of these decisions are made at the later stages of product development and have a limited effect on the total system. However, during the architecture development stage of a project the decisions made have a profound effect on product success.

An often-used heuristic is that the biggest mistakes are made on the first day. When a project is in its initial architectural phase of the design process many

decisions are made that are the basis for future decisions as the design matures. With these decisions the architecture provides the definition of interfaces between interacting components and provides structure for the product. Many times these decisions are made without full knowledge of the interactions between system elements. This lack of understanding can lead to poor decisions. Unfortunately, poor decisions made during the early stages of the program have a significant compounding effect. Once the architecture is frozen and product development teams are starting to create and implement the supporting design tasks, it becomes increasingly costly make significant revisions to the system architecture. Using Eppinger's [8] example from above, if a problem is discovered in the system architecture during the later stages of a design cycle, it might affect the work of up to 400 people. The impact can be mandatory design rework as well as delays in system verification and validation as a portion of the team has to delay activities while the others work on design recovery activities.

Since the initial architecture phase determines many of the important decisions about the structure and form of the product, it has been shown that the initial design stages are the major determinant of life cycle costs. For instance, Whitney and Nivens [21] suggest that about 70% of the life cycle cost of a product is determined at the conceptual design stage. Thus, in order to increase the probability of satisfying market requirements within the desired time to market window it is vitally important to develop system architectures that are "almost correct" at the start of the design process. One way to do this is to document the

relationship between market requirements and system components, as well as between components within the system, and use this information to generate the system architecture.

This thesis suggests a methodology using design structure matrix techniques in combination with a constraint satisfaction problem solver to develop the system architecture. This approach allows detailed evaluation of proposed architectures early in the product development cycle. Shortfalls in the proposed architectures can be revised via model-based iteration while identifying key areas where system knowledge is lacking. Management can direct the team to focus their attention on filling the knowledge gaps early in the development process and funneling the new information into the architecture model. The result is a product architecture that meets the known requirements, has improved stability, and contains the documented decisions that will be required for future analysis and review.

## **2 REVIEW OF PRIOR WORK – TOWARDS A FRAMEWORK**

The architectural phase of product development has received considerable attention in recent years as companies work to improve their capability to develop complex products. Many researchers have made significant contributions to understanding the complexity of product development. Others have focused their research in the more general area of understanding the dynamics of complex systems. In order to develop the framework that supports the proposed methodology it is useful to review the dynamics of complex

systems prior to reviewing the research in the development of complex products and systems.

According to Sterman [28], all learning and decision making is based on feedback. We make decisions that alter the real world, receive feedback, and use the new information to revise our understanding and decisions to bring the state of the system closer to our goals. This model of learning and decision making is closely aligned to the role of the systems architect and the decision-learning feedback cycle that occurs when a new system is being developed.

One of the problems with the decision-learning feedback process is that there may be a considerable time lag between when the decision is made and when meaningful feedback is received to enable learning. This is especially true in the case of architectural development on complex systems. From the time the architectural decisions are being made to the time that a system is operational might be several years. With a lag time of years, it is difficult to ensure learning. In addition, for the particular project at hand, the learning will do little good. The focus will be on how to solve any emergent problems in the shortest amount of time and not on the reflective learning. Any realizable benefits will be appropriated by the next generation of products.

Sterman [28, pg. 292] attempts to address these concerns and states:

“The challenge facing all is how to move from generalizations about accelerating learning and systems thinking to tools and processes that help us understand complexity, design better operating policies, and guide organization ...-wide learning... I argue that successful approaches to learning about complex dynamic systems require 1) tools to articulate and frame issues, elicit knowledge and beliefs and create maps of the feedback structure of an issue from that knowledge; 2) formal models and simulation methods to assess the dynamics of those maps, test new policies, and practice new skills; and 3) methods to sharpen scientific reasoning skills,

improve group processes, and overcome defensive routines for individuals and teams.”

Sterman proposes a learning model utilizing “virtual worlds” as a means to develop and test decisions prior to implementing them in the real world. This enables “on the job learning”, reduces the lag time between when a decision is made and when feedback is provided on the quality of the decision, and improves our understanding of the system. With refined understanding of the system behavior the system’s complex interactions become easier to manage. This model, shown in Figure 1, forms a framework by which abstractions such as modeling and simulation can be used to increase our ability to develop and manage the development of complex systems.

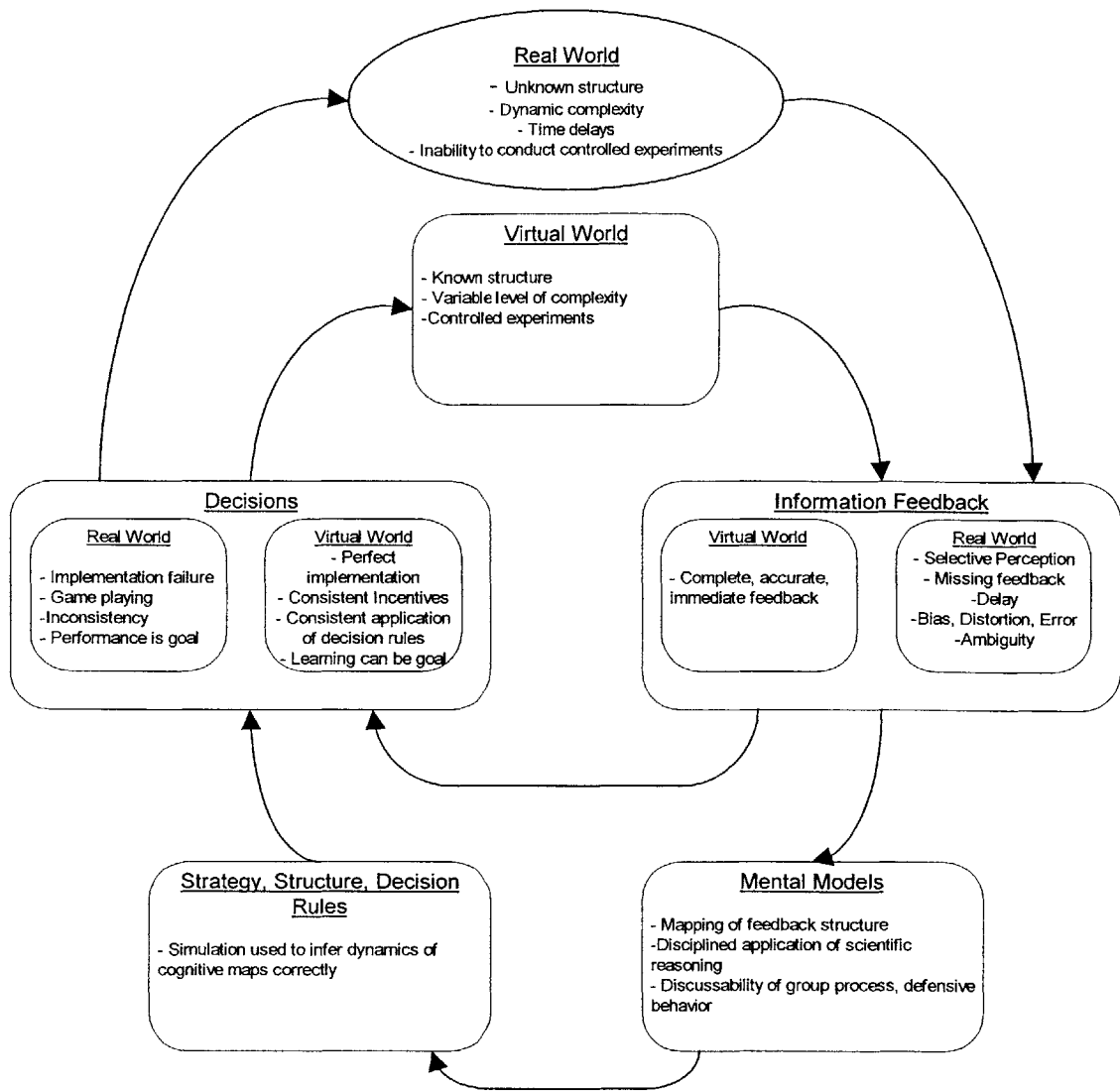


Figure 1 Sterman's Model for Learning

Sterman suggests a general framework for learning about complex systems. In this framework, the use of predictive models that represent the relevant parts of the system is crucial for both learning about the system and for promoting dialogue about discrepancies in the predicted performance with respect to the desired performance. This concept forms the basis for the development of the proposed methodology.

Modeling and simulation has been widely applied in the development and design of systems, including the initial architectural stages of product development. Several researchers have suggested methods and tools to improve the quality of product design at this stage of the product development cycle.

Jandourek [9] discusses product platform development and points out that, in traditional product development, the product architecture may be implicit and may not be written down. The architectural knowledge of a product is contained within the head of a single architect or small group of architects. Implicit and informal architectures are acceptable when team sizes are small and the level of complexity is low. However, as the level of complexity increases and team size and coordination issues become larger a more formal method is required. Formal architecture documents in the form of diagrams and text are required to make the system architecture explicit and enable others on the program to have access to the architectural knowledge they need to complete their design and implementation tasks. Having an explicit architecture also makes it possible to quantify trade-offs in a systematic way.

Yohannan [38] discusses the impact of increasing complexity on system development and concludes that with system complexity growing exponentially, even the most experienced system architects are hard pressed to anticipate and analyze all implications of different system architectures at the front end of the design process. Referring to the pre-concept and concept phase of a project as the front-end, the author also states that front-end errors can carry overwhelming

consequences at the back-end of the project. Tools that offer the ability to determine the architecture is right in the initial phases of design are in demand. The article goes on to describe an architectural development and evaluation tool suitable for the Internet-based products. This lends further support for the need to develop models of learning that support the creation and evaluation of system architectures.

Newbern and Nolte [22] in discussing the tension between art and science in systems engineering state that in their experience the most severe problems in developing large, complex systems originate from the “art-side”. They describe the art-side as making decisions on the high-level structure of the system in the absence of absolute knowledge and certainty. The art is making judgement calls based on limited information. They found that the most severe problems that arise later in the development process derive from early judgement decisions on the decomposition of the system. Again, the ability to increase the rate of system learning and being able to gain feedback on potential decisions via a model of learning before building the physical system would help to minimize these types of problems.

Thornton [31] has done considerable work in the area of key characteristics (KCs) and their use in successful product design. Key characteristic are defined as product, sub-assembly, part, or process features that significantly impact the performance, cost, or safety of a system. Variations in KCs outside of a specific range result in significant degradation in product performance. Thornton [32] has worked with several major design and manufacturing firms such as Xerox, GM,



Ford, and Boeing. All use some variation of a KC flowdown to identify the key characteristics. Thornton [31, p. 5] describes the KC flowdown process as:

“A KC flowdown is a hierarchy of variation-sensitive product requirements and part and process features that contribute to their variation... Product KCs are at the top of the tree and each product KCs has several contributing subsystem-KCs...”

According to this definition Product-KCs are variation-sensitive product requirements. These requirements are satisfied by the implementation of a unique collection of functional elements combined in the manner described by the product architecture.

According to Thornton [31 p. 9], in many cases, companies do not systematically flow KCs through the product level.

“Rather, they start by identifying product requirements; next, they identify, based on their engineering judgement, what individual features contribute to product requirements. Subsystem-KCs are often not identified. When a design team fails to create a systematic flowdown, too many KCs are identified, it can be hard to trace root causes (Lee and Thornton, 1996)...”

Although many companies are using key characteristics processes to identify and control sources of variation in their design, they are often failing on the first critical step in the process – identifying the linkage between product requirements and subsystem parameters. One of the outputs of an architecture development task should be to identify these linkages.

Based on the review of research presented thus far, it is possible to delineate some of the needs that an architecture methodology should satisfy. In particular the methodology needs to:

1. Provide a means for modeling the system that provides the capability to gain feedback on proposed decisions. This promotes rapid system learning.
2. Provide a definition of the linkage between product requirements and design parameters. From a key characteristic point of view, this creates a

map from product requirements (Product-KCs) to subsystem parameters (subsystem-KCs).

3. Provide documentation that makes the architecture explicit and enables others to have access to the architectural knowledge (decisions).
4. Increase confidence in the proposed system so that product design can proceed with a minimum of risk.

Two techniques provide capabilities that will help to achieve these goals. One of these is the design structure matrix (DSM) technique and the other is the constraint satisfaction problem (CSP) technique. The next section will briefly review these techniques and help establish the basis for their use in developing system architectures.

## 2.1 DESIGN STRUCTURE MATRIX TECHNIQUE

The design structure matrix (DSM) was first introduced by Steward [29] in 1981 as a framework for information flow analysis. A DSM consists of an N-square diagram showing the interaction of each element with every other element in the model. A sample DSM is shown in Figure 2.

	A	B	C	D	E	F	G
A			X	X			X
B					X		
C	X	X		X	X		
D		X					
E			X			X	
F							
G							

Figure 2: Sample DSM

The marks (X) indicate the existence and direction of information flow between elements in the matrix. Reading across a row, an X indicates information flow from the element in marked column. The element in the row is dependent on information from the element in the marked column and the element can not be

completed until the information is available. Reading down a column indicates which elements require information flow from the current element. In the example shown above, reading across row 'C' indicates that element C needs information from elements A, B, D, and E. Reading down column 'C' indicates that elements A, and E need information from element C. One of the properties of the DSM is that it can be used to highlight dynamics loops in the form of feed forward information flow and feed back information flow. To illustrate this point, again consider the example shown in Figure 2. Element C needs information from A, and element A needs information from element C. This forms an iterative loop. Element A can be completed by initially making an assumption about element C (for completeness an assumption also needs to be made about element D and G). When element A is completed, its output is used to revise the information for element C. Upon completion of element C, its information is again used to iterate element A whose output is used to revise element C, and so on. The number of iterations required depends upon the nature of the problem.

Using the nomenclature adopted by Steward [29], elements below the diagonal represent the forward flow of information (feed forward). Elements above the diagonal indicate a feedback of a later element to an earlier one. Since the DSM is a matrix structure, matrix operations can be performed on it. By using matrix manipulation algorithms, it is possible for the user to restructure the order of elements in a matrix.

There are two types of restructuring algorithms that are commonly used in a DSM analysis. The first is partitioning whose goal is to render the matrix lower

triangular to the extent that it is possible. This minimizes the feedback loops, which are represented by marks above the matrix diagonal. A structure without feedback loops is typically easier to understand and implement. Partitioning is typically used to eliminate iteration cycles in decision structures and project plans. It provides a structured flow of information in a project. Figure 3 shows how partitioning would change the matrix shown in Figure 2. The elements have been reordered such that element F is now the first element that should be completed and then element G. Also the partitioning has concentrated the iteration loops such that there are two nested loops consisting of elements A, B, C, and D in one loop and elements B, C, D, and E in the second.

	F	G	A	B	C	D	E
F							
G							
A			X		X	X	
B							X
C			X	X		X	X
D				X			
E	X				X		

Figure 3 Partitioned DSM

The second restructuring algorithm is known as clustering where the matrix is manipulated to produce tightly coupled feedback loops. The goal of clustering is to find subsets of DSM elements (i.e. clusters or modules) that are mutually exclusive or minimally interacting subsets. The intent of clustering is to identify the set of elements in which the internal relationships have a high level of interaction, while external interactions are eliminated or at least minimized. Clustering is typically used for decomposing organizations into logical teams, but can also be used to organize systems into subsystems and modules. Figure 4

shows the effect of clustering on the DSM from Figure 2. The restructured matrix has been clustered into four groups. These clusters have a high level of internal interactions and low levels of external interactions. These clusters could be product development teams, or product subsystems.

	A	D	C	E	B	F	G
A	-	X	X				X
D		-			X		
C	X	X	-	X	X		
E			X	-		X	
B			X	X			
F						-	
G							-

Figure 4: Clustered DSM

During the different stages of the architectural development phase, there is a need for both types of restructuring. Clustering is appealing in that the creation of tightly knit modules (and teams) enables simpler system interfaces. Complexity resides within the modules not in the interfaces. Partitioning enables better organization of development tasks and helps to identify which elements need immediate attention.

During the past few years, DSM has received significant attention as a means to organize and structure complex systems. These applications have tended to concentrate on the decomposition of organizations into team structures and systems into the appropriate “design” chunks [35, 24 12, 27].

Steward [30] provides a simplified example of the application of DSM to the design of an electric car. The first step is creating a list of the pertinent tasks. The tasks are the set of activities required to develop and implement the design such as the determination of design variables, the execution of design tasks, and

the preparation of documentation. In the second step, the precedence relationships between the tasks are documented in the DSM structure, including a measure of the strength of the relationship. Finally, the matrix is partitioned to order the tasks to reduce iterative coupling.

Pimmler and Eppinger [24] suggest a three-step methodology to define the architectural chunks. These steps are (1) decompose the system into to elements, (2) document the interactions between elements, and (3) cluster the elements into chunks. The DSM technique is used to document the interactions and to reorder the elements into architectural chunks using clustering.

They suggest that the system decomposition be completed such that the elements are specified to one level of detail further than that desired for the product architecture. In addition, they note that functional and physical elements can be used in the decomposition. They suggest that in a clean-sheet design situation one would primarily utilize functional elements because most of the physical elements would not have been identified. In a variant design situation, where physical design elements are available and well understood then they would be used in the decomposition. In many cases, a mixture would mostly likely be used. The example discussed in [24] creates a DSM using the physical elements of an automobile climate control system. In general, DSM has been used very successful to capture the interactions between elements and enable the ordering of the interactions to be optimized for a given problem domain.

The design structure matrix technique has been shown to be a very powerful tool in ordering and reducing the complexity inherent in systems with numerous tasks. Other researchers have suggested similar techniques using network diagrams to capture dependencies. For example, Fujita and Shinsuke [5] propose an agent-based distributed design method for complicated, large engineering systems. They illustrate their method using examples from ship design. Their methodology identifies dependency and concurrency in design activities, which are classified into the following activities: generating system structures, configuring spatial relationships, determining system attributes, and selecting system components. Their method uses the dependencies between objects to develop the system architecture.

In another example, Kusiak and Wang [13] propose a methodology to assist designers in negotiation of constraints. They propose a network model to represent relationships among design variables and an algorithm is developed to derive dependencies between design variables and goals. They illustrate the methodology using an example of a spring design in a mechanical system.

Other techniques have been developed that to ease the task of system design by explicitly documenting the relationship between customer requirements and design parameters. One such method is the Quality Function Deployment (QFD) process. The process was developed to improve the alignment between customer-requested attributes and the design implementation.

The QFD technique uses a series of matrix-like structures to systematically map the requirements (referred to as the "WHATs") to the product engineering

elements (referred to as the "HOWS"). The "WHATS" are listed in the first column of the matrix and the "HOWS" are listed in the first row. Relationships are indicated at the intersection of the columns and rows. In addition a triangular matrix is placed on top the "HOWS" and is used to indicate relationships between the various "HOWS". This gives the QFD matrix a house-like structure. QFD is sometimes referred to as the "house of quality".

QFD has been successfully used to capture the relationship between customer requirements and product design variables. A multi-stage "house of quality" process has been used successfully to establish the relationship between customer requirements and product-KCs, and from product-KCs to part-KCs. For example, Portanova, and Tomei [25] discuss the use of the QFD process in launch operations in a paper prepared for the Space Systems Division of Air Force Systems Command. The report describes the application of the QFD methodology to launch operations with a goal of developing a more efficient launch capability that is more reliable and lower cost than current systems. The paper emphasizes that QFD enables the development team to systematically analyze the customer requirements and identify linkages between the critical characteristics that determine an efficient launch system capability.

#### DSM VERSUS QFD

The QFD process captures much of the same information as a DSM. Although some versions of QFD capture competitive benchmarking data and other related information.



A valid question is “Why use the DSM technique when other techniques have been developed specifically for capturing the linkage between customer requirements and design parameters?” Put simply “Why not use the Quality Function Deployment (QFD) process?”

For the architectural development framework being developed in this paper, I believe that the DSM technique offers several advantages over the QFD technique. The biggest advantage is that the DSM is a matrix structure and is therefore amenable to matrix analysis. As mentioned earlier, the DSM can be re-structured using matrix operations. This presents several opportunities:

- ☑ The relationships can be easily reorganized to communicate efficiently with diverse groups. Partitioning and clustering provide different views of the system. In addition, the architecture DSM can be used to suggest organizational team structures.
- ☑ The DSM can be used to drive the development of the CSP model in an organized manner. Large programs become difficult to manage and the structure becomes unorganized as changes are made to the original program. The DSM technique can be used to suggest “modules” for the constraint-based model.

In addition, Smith and Eppinger [27] developed a process by which DSM relationships can be quantified and analyzed using eigenvalues. This analysis allows for a deeper understanding of the relationships within the DSM. Determining the eigenvectors and eigenvalues of the modified form of a DSM enabled Eppinger and Smith to identify the areas of the design problem that would require lengthy iteration to reach a technical solution. They were also able to identify the rate at which iteration occurred in the project loops. This capability can be applied to architectural modeling as a means to judge the impact of potential changes to the design. As the project progresses the model can be

used to guide the project management team in making informed choices about seemingly equivalent changes.

In the author's opinion, the DSM technique offers several advantages over and above those offered by the QFD technique. Those advantages can be exploited to improve the development of system architecture and improve the communication of architectural decisions.

One of the concerns with the DSM technique is that although the relationship between elements can be identified in the DSM matrix, it is difficult to capture the exact form of the dependency. For example consider the DSM shown in Figure 5 which represents the dependencies between print speed (pages/sec), photoreceptor length (mm/rev), process velocity (mm/sec), and the number of

	Process Speed	Number Pitches	Photoreceptor Length	Print Speed
Process Speed				
Number Pitches				
Photoreceptor Length		X		
Print Speed	X	X	X	

Figure 5: DSM Example for Print Speed

pitches on the photoreceptor (prints/rev).

Although the DSM structure shows that Print Speed has a dependency on the other three elements, it does not show the form of the dependency. The arithmetic relationship is shown below.

$$\text{Print Speed} = \text{NumberPitches} \times \left( \frac{\text{ProcessVelocity}}{\text{PhotoreceptorLength}} \right)$$

In order to create a comprehensive framework, a complementary technique is needed - one that can capture and use the detailed relationship between elements to enable architectural development. One possible approach is to use constraint satisfaction techniques.

## 2.2 CONSTRAINT SATISFACTION PROBLEMS

Design problems can be thought of in terms of satisfying a set of constraints. An acceptable design would be one in which all of the constraints are successfully satisfied. In this light, design activities are focused on solving the constraint satisfaction problem.

A more formal definition of a Constraint Satisfaction Problem (CSP) is a problem in which the problem variables are described by a set of variables each with a finite domain of values and a set of constraints on these variables. The discrete set of values that satisfy the constraints on the set of variables is a solution to the CSP. Using Constraint Satisfaction Problems methods to develop a design solution is frequently referred to as constraint-based design.

Thornton [33] describes the use of constraint-based design for embodiment design. Embodiment phase of design is defined as the process by which design concepts are given geometric form. She proposes that one way to characterize mechanical design is to describe it as a process of constraint specification and satisfaction. She describes the development of a computer tool to aid in constraint specification and satisfaction of mathematical constraints for the embodiment design phase of mechanical design. The intent of the tool is to aid

designers in the development of detailed level designs. The design objects available in the tool are lower level components such as shafts, pins, and gears. Although the author discusses constraint-based design for use in detailed design, it can also be applied at higher levels of abstraction.

Charman [2] describes a knowledge-based system that generates all possible floor plans satisfying a set of geometric constraints on the rooms. The constraints, which are derived from the architect's specifications, describe boundaries on acceptable room floor plan architectures such as non-overlap, adjacency considerations, minimum/maximum area, and minimum/maximum dimensions.

System architectural constraints are derived from a multitude of sources.

Some of the possible sources are:

- Product Requirements – These are usually the first and highest level of constraints. They specify what the design must ultimately accomplish.
- Object Technology - The chosen technology set imposes another layer of constraints at the functional level. Technologies need to be combined in such a way that they perform the desired system functions.
- Object geometry and relationships – This is the preferred physical embodiment of the technology set. In order to achieve the desired functionality geometric components need to be properly oriented. This produces constraints for position/location, absolute orientation and orientation relative to another object.
- Serviceability/Manufacturability – The requirement that the system be manufacturable and serviceable places constraints on the architectural design. An example of a constraint would be minimum clearances between modules to ease installation and removal.

In order to be solved with the aid of a computer, the constraints need to be expressed mathematically. The expression will take the form of equalities, inequalities, and conditional statements.

One of the drawbacks of the constraint-based design approach is that the design variables and constraints tend to be encapsulated in a programming environment. Communication and documentation of the linkages is done at a detailed level and the constraints are typically expressed in terms of mathematical formulations. This makes communication difficult and cumbersome. It is difficult for a person who is not associated with the effort to grasp the detailed relationships and be able to equate the constraints to the proposed architecture.

The use of the DSM technique to capture and communicate architectural dependencies and the constraint-based model technique to develop a model capable of proposing suitable architectural designs provides the foundation from which the architecture development methodology will be created.

### **3 IMPLEMENTATION OF PRACTICAL METHODOLOGY**

In the previous section, it was noted that an architecture methodology should satisfy the following needs:

1. Provide a means for modeling the system that provides the capability to gain feedback on proposed decisions. This promotes rapid system learning.
2. Provide a definition of the linkage between product requirements and design parameters. From a key characteristic point of view, this creates a map from product requirements (Product-KCs) to subsystem parameters (subsystem-KCs).
3. Provide documentation that makes the architecture explicit and enables others to have access to the architectural knowledge (decisions).

4. Increase confidence in the proposed system so that product design can proceed with a minimum of risk.

In this section, a general methodology for developing system architecture that is capable of satisfying these needs is described. Before discussing the methodology, it is important to discuss the context in which the method is intended to be used.

### 3.1 PROCESS CONTEXT

The development of system architecture has to be an integral part of the product development process. Figure 6 shows a schematic of a product development process.

Architecture development starts in the Pre-concept phase and is completed during the Concept phase. The Product Strategy phase provides the product and market requirements that define the goals that the system must meet. The system architecture must be designed such that the resultant system will satisfy those goals.

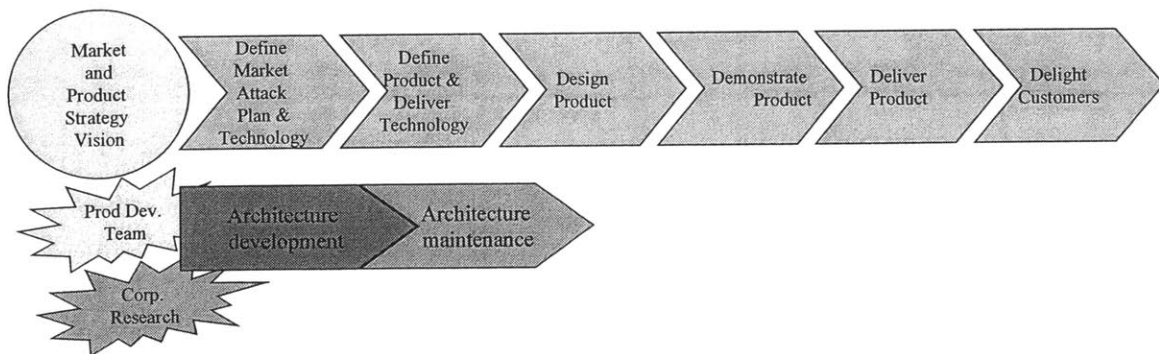


Figure 6: Product Delivery Process

As shown in Figure 7 [15], the market requirements are mapped to goals and then to system functions. The elements on the left are the upstream influences and are typically captured in the Product Strategy documentation. The elements on the right are some of the more important downstream influences. These constraints are placed on the systems by manufacturing capabilities and serviceability requirements.

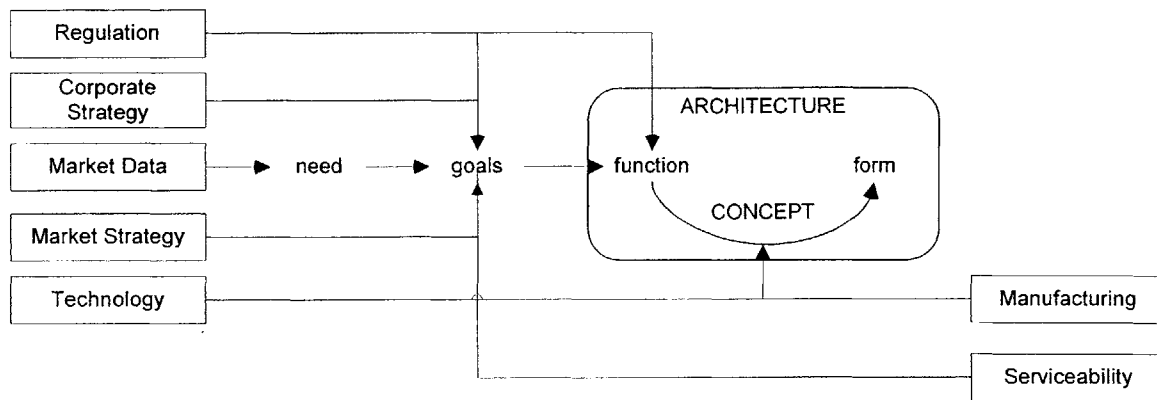


Figure 7: Influences on System Architecture

The influences will be transformed into the dependencies, captured in the design structure matrix, and form the basis for the constraints in the constraint-based design. These techniques will be used to create the concept that maps the required functions into the physical form.

### 3.2 ORGANIZATIONAL CONTEXT

The product development process is tightly linked to its product delivery organizational structure. In the case study discussed in section 4, two organizations play an important role in the development of the system architecture. The corporate research group develops the core technologies for printer products, and product development teams are responsible for system

development and refining the subsystem technologies for use in a particular application.

The need for future printing capabilities is identified as part of the Product Strategy activities. The corporate research groups are responsible for developing suitable subsystem technologies that will satisfy future market needs.

As the technology matures, the product development teams work with the research groups to transition the technology into the product. It is at this point that the system architecture is defined in enough detail to enable the product development team to design a production system.

### 3.3 METHODOLOGY DESCRIPTION

In many ways, this approach is an integrative approach to product architecture. Subsystem technologies are combined to compose a system that satisfies a set of product requirements. However, market needs change from the time that the initial need for new technologies are identified and when the product development team begins development of a specific product. Because of these changes, it is not uncommon for revised market requirements to require changes in the subsystem technologies. For example, the market might have matured or a competitive offering might have emerged such that a higher print speed is now needed to satisfy market requirements. This requires that the subsystem technologies be able to operate reliably at a higher speed. Thus, the integrative approach becomes more complex than simply putting together pieces of a puzzle. Oft times, the size and shape of the pieces change at the time when they are being put together.



With these concerns in mind, Figure 8 shows the process steps for an integrative architecture development methodology.

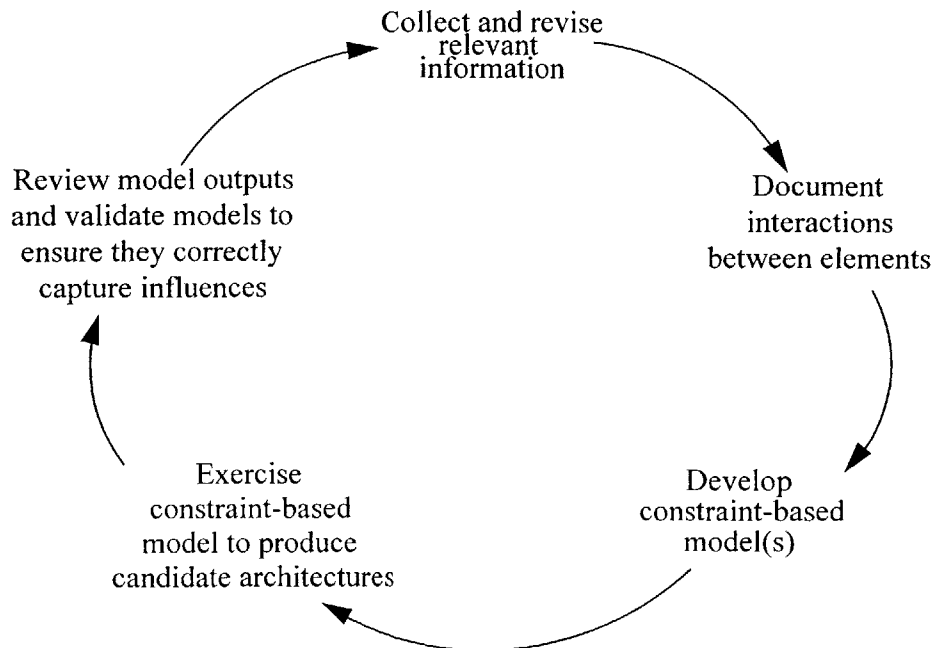


Figure 8: Process Flow for Integrative Architecture Development Methodology

As the teams use the models to develop a deeper understanding of the relationships embodied in potential architectures, they will need to loop back to previous steps to make sure that the models capture their understanding. The remainder of this section reviews the major elements in each step of the methodology.

#### COLLECT THE RELEVANT INFORMATION

During this step all requirements that need to be considered are pulled together.

This will include product requirements, technology selections, and design and manufacturing concerns. Based on prior experience, it is recommended that

one-on-one meetings be held with each contributor. This is especially important for understanding the theory of operation behind the new subsystem technologies.

#### DOCUMENT INTERACTIONS BETWEEN THE ELEMENTS

During this step, the initial DSM is created from the information gathered in step one. In addition, reviews with research teams and product development teams are required. These should be a combination of one-on-one reviews and formal design review meetings. To further verify requirements and linkages, it is sometimes beneficial to begin the development of the constraint-based model. The detailed nature of the constraint-based model helps to provide a clearer understanding of the relationships between elements. The output from this step is the initial documentation of architecture relevant key characteristics, subsystem design rules of thumb, physical constraints, and system requirements.

#### BUILD CONSTRAINT-BASED MODELS

During this step, the dependencies are translated into a format acceptable for the constraint-based modeling system. This requires that the key characteristics nominal and ranges, design rules, and requirements be converted into algebraic constraints. Typically, these take the form of equalities, inequalities, and logical assertions.

It is important to work with the subsystem teams to ensure that this translation is acceptable and maintains the intent of the given design rules. The geometric model is linked with the functional constraint model to produce the constraint-based model.

#### EXERCISE CSP MODEL TO PRODUCE CANDIDATE ARCHITECTURES

The constraint-based model will use the requirements and dependencies to develop architectures that satisfy all the constraints. The output is a high-level design for the system.

#### ENSURE MODELS CORRECTLY CAPTURE INFLUENCES

This is the single most important step in the process. After the constraint-based model is producing reasonable results, it is critical to go back to the original sources of the requirements and validate that the model correctly represents the intent of the requirements. One way to do this is to have an architectural review meeting on a regular basis. This allows for a group discussion of the formulation of the architecture and helps to share the rationale for decisions. It also provides a forum to discuss trade-offs and prioritization between requirements. This is a source of information feedback Sterman [28] discusses in his model of learning.

### 3.4 TOOLS TO SUPPORT METHODOLOGY

This section discusses applications and tools that support the methodology. Two tools are needed for this approach, a DSM tool and a constraint-based modeler. This section is not intended to be an exhaustive discussion of all applications in these areas of study. Instead, it is included to provide a practical assessment of the tools that the author has used and to provide a starting point for others interested in this approach.

#### DSM TOOLS

Several tools automate the creation and manipulation of a DSM.

Problematics offers a Windows-based application, and Jim Rogers of NASA has developed DeMAID that runs on many Unix platforms as well as the Apple

Macintosh. MIT has developed a set of macros to enable DSM creation and manipulation in Excel [17].

Although any of the tools can be used to create the DSM, the following observations can be noted. DeMAID is more suited towards task or activity based analysis. Problematics uses tearing algorithms and does not support clustering. The MIT Excel macros support tearing, partitioning and a clustering algorithm will be available in the near future [17]. Fortunately, the data can be shared between the applications using the CSV file format.

#### CONSTRAINT-BASED MODELER

The constraint-based modeling system used in the case study described in section 4 is Mechanical Advantage by Cognition Corp. It is part of the modeling environment used by mechanical engineering and design groups at several large companies.

Mechanical Advantage is a comprehensive mechanical modeling environment that allows an engineer to model the function and performance of a design concept. At its core is an intelligent sketcher that can be used to capture design intent including all geometry and geometric constraints. A design coordinator allows the engineer to build a complete product design model by linking the geometry/constraint model to the performance model (equation sets), empirical data sets, external programs, tolerance models, documentation models, and manufacturing process models [3].

The Mechanical Advantage application is well suited for a constraint-based approach to architecture. The combination of flexible geometry creation with geometric constraints, and the ability to link geometric objects to algebraic and

logical “design rules” provides many of the capabilities that are needed to develop a constraint-based model. The availability of this type of tool is critical for this methodology.

Figure 9 shows a simple example created in the Mechanical Advantage environment. A rectangle is created in the Sketch Note and an algebraic equation in a Math Note is used to constrain the length of side B to be twice that of side A. In this example, the engineer can specify side B and the system calculates side A. User controlled variables have padlock icon next to them, while a calculated variable has a  $\therefore$  symbol next to it.

In the example, the variables for the length of side A and B are linked to dimensions of the rectangle. Links are represented by a  $>>$  or a  $<<$  symbol. The  $<<$  indicates that the link originates from this variable and the  $>>$  indicates that the link is going to this variable. The area, which is a property of the rectangle, is linked from the rectangle’s physical property sheet to the Area variable in the Math Note.

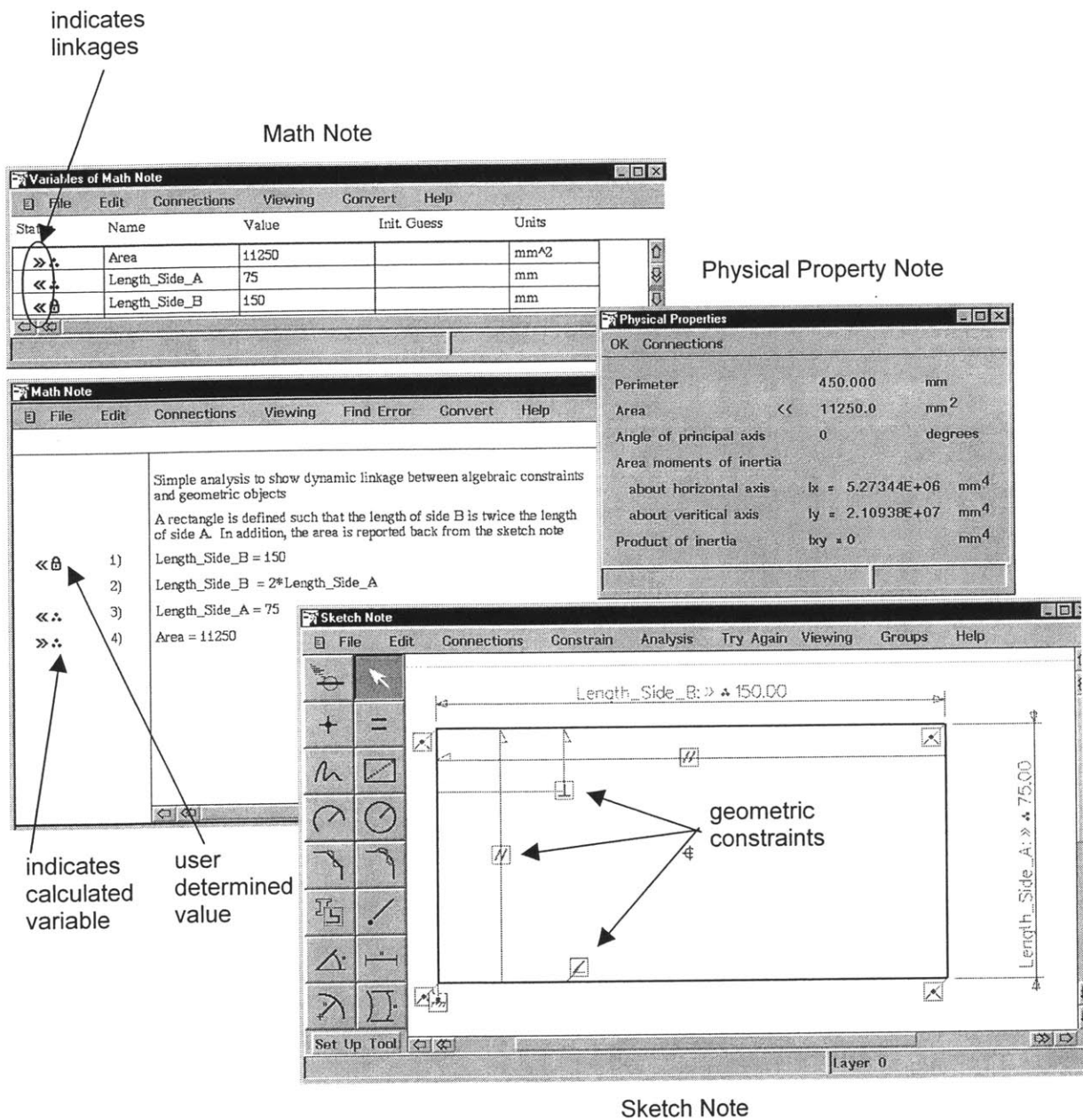





Figure 9: Example in Mechanical Advantage

The Sketch Note also shows the geometric constraints of the rectangle. The rectangle is constructed by having the ends of the sides be coincident with other sides. This is represented by a  symbol. The opposite sides are constrained to be parallel, which is represented by . Two adjacent sides are constrained to be perpendicular. This constraint is represented by . Besides the two linear dimensions, two other geometric constraints are required to fully constrain the rectangle. The first one is a fixed-point constraint and the second is a fixed angle constraint. These are used to fix the location and orientation of the rectangle within the Sketch Note.

### 3.5 ROLE OF THE ARCHITECT

The last concern in the methodology is to discuss the role of the architect. In the proposed methodology, the architect plays the role of integrator. As noted above, given a product development process that identifies key technologies in advance of actual product embodiments, the development of product or platform system architecture is integrative in nature. The architect's job is to match technology capabilities with market requirements. This section discusses some of the important decisions that the architect needs to make.

One of the essential roles of the architect is to determine the boundary of the activity. This is true for the physical domain as well as for the models being developed to guide the architect. These decisions can have a significant impact on the scope and complexity of the DSM structure and the constraint-based model. The two decisions although tightly coupled can be made in a quasi-independent manner if the boundary and therefore the scope of the physical

system architecture are determined first. Then the boundary of the supporting models can be selected.

Once the boundary for the physical architecture has been determined, the architect can start to consider the working boundaries for the supporting models. The decision on what to include and how to include it depends on technical complexity and organizational relationships. The architect must also determine the form of the model and the level of detail at which the model will be implemented. Each of these decision points will be briefly discussed in the following sections.

#### FORM AND LEVEL OF DETAIL OF MODEL

After defining the boundary for the system, the next consideration is to decide the content of the models. This is an important consideration. Since subsystem technologies have already been developed to at least a partially functional level, there is a good chance that analytical models exist for some of the subsystems. The same is true for certain system models. For example, in the case study discussed in section 4, the company's system image quality model has been developed over the past twenty years and captures much of the physics involved in electrophotography. The question the architect must face is whether the architecture model should duplicate existing models and to what extent should there be analysis overlap. Figure 10 illustrates the issue. The large ellipse represents the boundary of the physical domain that the architect is working in. The small ellipses inside the boundary represent the various models that contain information relevant to the creation of the system architecture. The models can be subsystem, functional descriptive models such as a model



predicting voltage uniformity on a photoreceptor due to the configuration of a charge device. The models can also be structural analysis models such as FEA, which determine the minimum required size for physical elements. System models have been developed, sometimes in parallel with the architecture model and sometimes as part of long-term development effort, that spans several product generations.

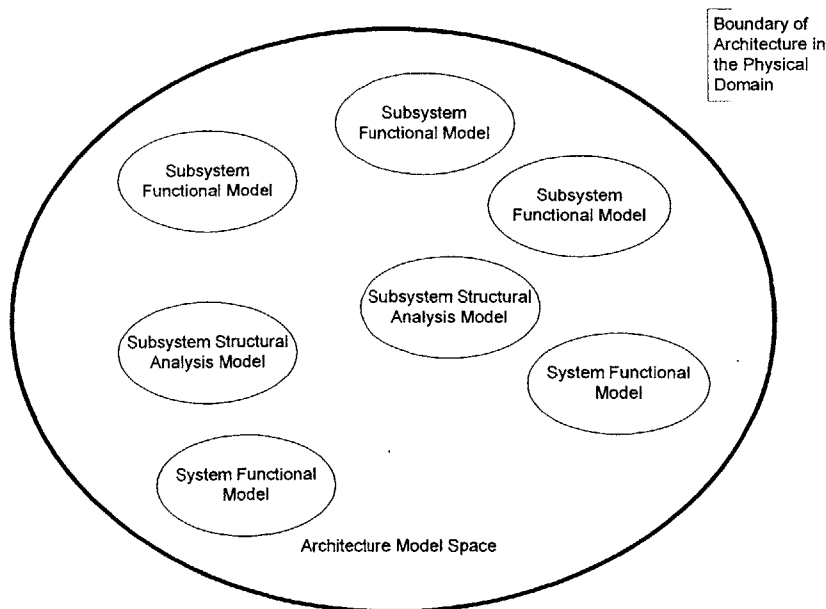


Figure 10 Physical Domain and Model Boundaries

Clearly one would not want to duplicate all of the capabilities in existing analysis. Instead the architect needs to understand what is the minimum amount of information required to complete the architectural model and negotiate with the subsystem teams on when and how relevant information will be shared.

This raises an interesting point in terms of key characteristics. In order to avoid duplication of efforts, it seems reasonable that the information included in the architecture model be an output from the subsystem functional analysis. In

many cases, these will be the physical attributes of the subsystem design. In such a case, the architecture model will include the subsystem's part-KCs.

For cases in which there is not an existing subsystem analysis, it is appropriate that the architecture model include the functional analysis of that subsystem. The architectural model might include a module to calculate the parameters needed to represent the subsystem. Thus, the model will include subsystem-KCs.

In an actual product development environment, the architecture model contains product-KCs (requirements), subsystem-KCs, and part-KCs. This is similar to the finding reported by Pimmler and Eppinger [24] that functional and physical elements can be used in the decomposition. The additional burden in this methodology is that since a high level design is produced by the constraint-based model, it must include the physical elements of the system. This suggests the use of distributed subsystem models with an integrative system model.

#### TECHNICAL COMPLEXITY & TECHNICAL EXPERTISE

Similar to the concerns raised in the previous section, a complementary concern is one of architect's technical expertise and the complexity of the system. For a simple system, it is possible that a single person could understand the interactions and decide how to create acceptable system architecture. In others words, it is possible that the architect would have the required technical expertise to understand the interactions and trade-offs. However, in a complex system, especially a clean sheet product with many new technologies, it is unlikely that anyone understands all of the subsystem and system level interactions. Therefore, the development of the system architecture becomes a

team effort with the architect acting as the central integrator of information. The distributed subsystem with an integrative, system architecture, model structure discussed in the previous section fully supports this approach to architecture development.

#### ORGANIZATIONAL RELATIONSHIPS

Using the product development process, described in Figure 6, and the description of the organizational structure, it is apparent that the architecture development phase of a project requires coordination of the organizational relationships as well as the technical relationships. In fact promoting open, honest communication and creating an environment in which people are willing to share their insights, is possibly more important than any single technical detail. One of the roles of the architect is to help to promote this type of an environment. It should also be pointed out that the fostering of open communication needs to be one of the primary objectives of the project management team. The distributed subsystem with an integrative, system architecture, model structure discussed in the previous sections can be used to create a more open environment. This approach prevents the creation of an all powerful system model that tends to stifle communication as opposed to embrace it.

In order to create this type of an environment, the following steps are recommended:

- Multiple one-on-one meetings with important technology and product development stakeholders. Discussions should center on the architectural implications of using a given technology set. The intention is to build a relationship that encourages open dialog. Agreement should be reached on the type and frequency of information exchange. The following table lists some of the information that needs to be discussed.

Technology Checklist	Subsystem Design Checklist
<ol style="list-style-type: none"> <li>1. Description of technology</li> <li>2. Guidelines for technology implementation</li> <li>3. Technology scaling rules</li> <li>4. Alternate technology options and technology breakpoints</li> <li>5. Description of technology development models and agreement on their use to support the architecture development effort.</li> <li>6. Description of key characteristics that influence the architecture</li> <li>7. Acceptable nominal and ranges for key characteristics as well as limits at which the technology will no longer meet its function, even with additional investments.</li> <li>8. Explicit agreement on the relationships to be represented in the DSM</li> <li>9. Explicit agreement on the mathematical expressions to be used in the constraint-based model</li> </ol>	<ol style="list-style-type: none"> <li>1. Description of subsystem interfaces</li> <li>2. Description of design implementation, this is especially important for assemblies that embody multiple functions.</li> <li>3. Description of CAD models and agreement on their use to support the architecture development effort.</li> <li>4. Description of key characteristics that influence the architecture</li> <li>5. Acceptable nominal and ranges for key characteristics.</li> <li>6. Explicit agreement on the relationships to be represented in the DSM</li> <li>7. Explicit agreement on the mathematical expressions to be used in the constraint-based model</li> </ol>

- Regular architecture design reviews. All interested parties need to be invited and given a chance to provide input. Unlike the one-on-one meetings that focus on a specific subsystem, this meeting focuses on the system and the interaction between competing requirements and design rules. The system models are used to focus the conversation on the key problem areas.

### 3.6 METHODOLOGY FLOW DIAGRAM

Figure 11 provides a graphical overview of the methodology.

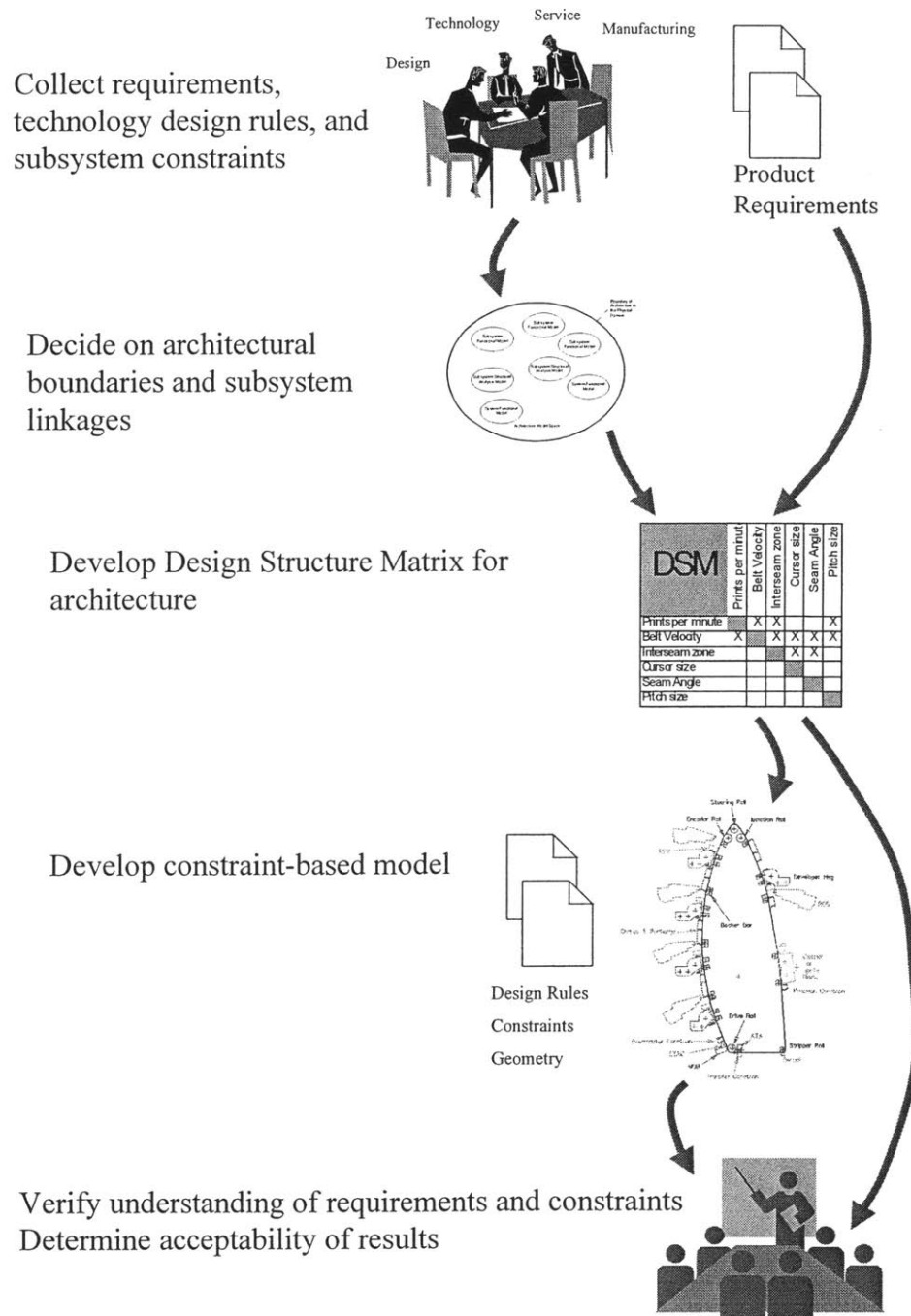


Figure 11: Methodology Flow Diagram

## **4 CASE STUDY -- XEROGRAPHIC COLOR ARCHITECTURE**

### **4.1 CASE INTRODUCTION**

This section of the paper discusses the use of the proposed methodology on a digital printer. The project is a clean sheet design for a high volume, digital color press. The case study will focus on the development of the xerographic module architecture.

In early 1994, I was given the opportunity of being on the Product Architecture Assessment Team for a new digital color printer and participated in the development of the architecture of the xerographic module. The case study covers a four-year time span during which the product transitioned from the pre-concept phase to the detailed design phase.

The xerographic architecture was successfully developed using a constraint-based model using Cognition's Mechanical Advantage. The DSM technique was not used on the project. The DSM technique was added to the methodology based on the author's experience as the architect/integrator of this xerographic module. The DSM constructed for this paper has been reviewed by others involved in the development of the product. Whereas the comments and insight into the value of the constraint-based model technique are based on first hand experience, the discussion on the use of DSM is necessarily based on hindsight.

### **4.2 ARCHITECTURAL PARTITIONING**

The high speed, digital printer is partitioned into four modules: the digital front end, input devices, output devices, and the print station. The digital front end is responsible for decomposing and preparing a client's job for printing. The print station creates the printed sheets for the job. The input devices provide cut

sheet media to the print station, and the output devices produce completed documents from the printed sheets in the format requested by the client.

The print station is decomposed into two major mechanical modules: the paper path and the xerographic module. The paper path modules include all the subsystems that are required to register, fix, duplex and otherwise move the media through the machine. The xerographic module includes all subsystems required to create a physical image and transfer it to the media.

#### 4.3 XEROGRAPHIC MODULE

Before discussing the application of the methodology, it is worthwhile to provide an overview of the xerographic process. The overview will discuss monochrome xerography. The discussion will then be extended to include the further complications of color xerography. This will help to establish the technical framework for the rest of the case study.

##### OVERVIEW OF XEROGRAPHY

The key component of xerography is the photoconductor. A photoconductor, also referred to as a photoreceptor, changes charge states when exposed to light. The xerographic process is based on this phenomenon.

Figure 12 shows a typical xerographic module for a monochrome printing system. The important steps in the process are:

1. A charge device is used to create a uniform charge on the photoreceptor.
2. The photoreceptor is then exposed using a source of light, such as a laser, creating a latent image. In most digital printing systems, exposing the photoreceptor causes it to discharge.
3. The photoreceptor with the latent image is then exposed to a field of charged particles, called toner, that is charged so as to be attracted to the latent image

on the photoreceptor. This stage is typically called development in reference to the similar step in photography when the image becomes visible.

4. The photoreceptor now has a fully developed image on its top surface consisting of charged toner particles. The next step is to pre-treat the image with a charged field to prepare it for transfer to a piece of paper. The pretransfer treatment reduces the strength of the charge holding the toner to the photoreceptor.
5. A piece of paper with a charge that will attract the toner from the photoreceptor is brought into contact with the photoreceptor. The toner is transferred to the paper, which continues to the fixing subsystem, which melts the toner into the paper.
6. The photoreceptor is subjected to a cleaning cycle that removes any residual toner.
7. The cycle is repeated for the next image.

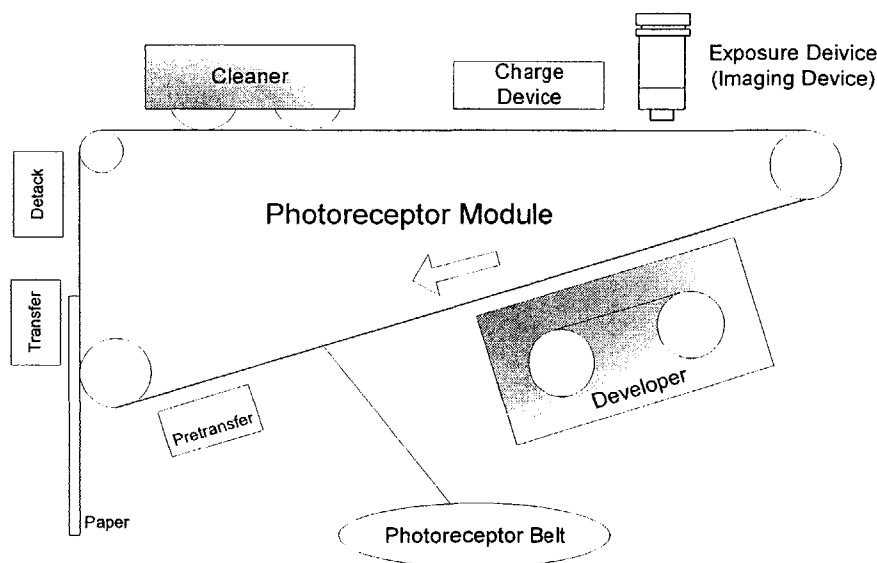


Figure 12: Monochrome Xerographic Engine

#### ADDITIONAL CONCERNS WITH COLOR XEROGRAPHY

Color xerography is more complex than monochrome xerography. A color xerographic image is created using multiple layers of toner. Subtractive colorants, typically magenta, cyan, and yellow, with the addition of black are used as the base colors for the system. Multiple imaging sequences are needed to



build the color image from the base colorant toners. These systems require more parts with more complex interactions. There are three predominant designs that are used to create a colored image.

#### Recirculating architecture

The earliest color xerographic products used a recirculating architecture. In this design, multiple revolutions of the photoreceptor are required to create a color image. Multiple development stations, each containing a different color, are sequentially positioned against the photoreceptor and the image for that color separation is developed and transferred to the paper. The sequence is repeated until all color separations have been imaged and transferred to the paper.

Productivity is one-fourth the productivity of the comparable monochrome system, Image quality is affected by the ability to maintain image registration while repeatedly aligning the paper to the next image.

#### Tandem architecture

Tandem color architecture has multiple xerographic engines arrayed such that each engine prints a single color image. The images are combined either on an intermediate substrate or directly on the paper.

Productivity is 100% for a color image. The drawback is a greater number of parts and therefore a greater concern for reliability and total cost of ownership.

#### Single pass multi-color architecture

The latest approach to color xerography is to create the full color on the photoreceptor and transfer the complete image to paper. This requires that the color image be built on the single image panel by going through multiple

xerographic stations. In this implementation, the term xerographic station is used to describe the set of charge devices, exposure device, and development devices required to produce a single color image. In a four color system there would be four xerographic stations - one for each color. The system is often referred to as an image on image system. Refer to Figure 13 for a schematic view of the architecture [23]. Appendix A describes the system in more detail. Similar to the tandem architecture the productivity is 100% for a color image, no recirculation is needed create a full color image. One of the key architectural concerns for development of this system is the management of the allocation of space around the photoreceptor module. The architecture shown in the figure was developed as part of this case study. Its development will be discussed in the following sections.

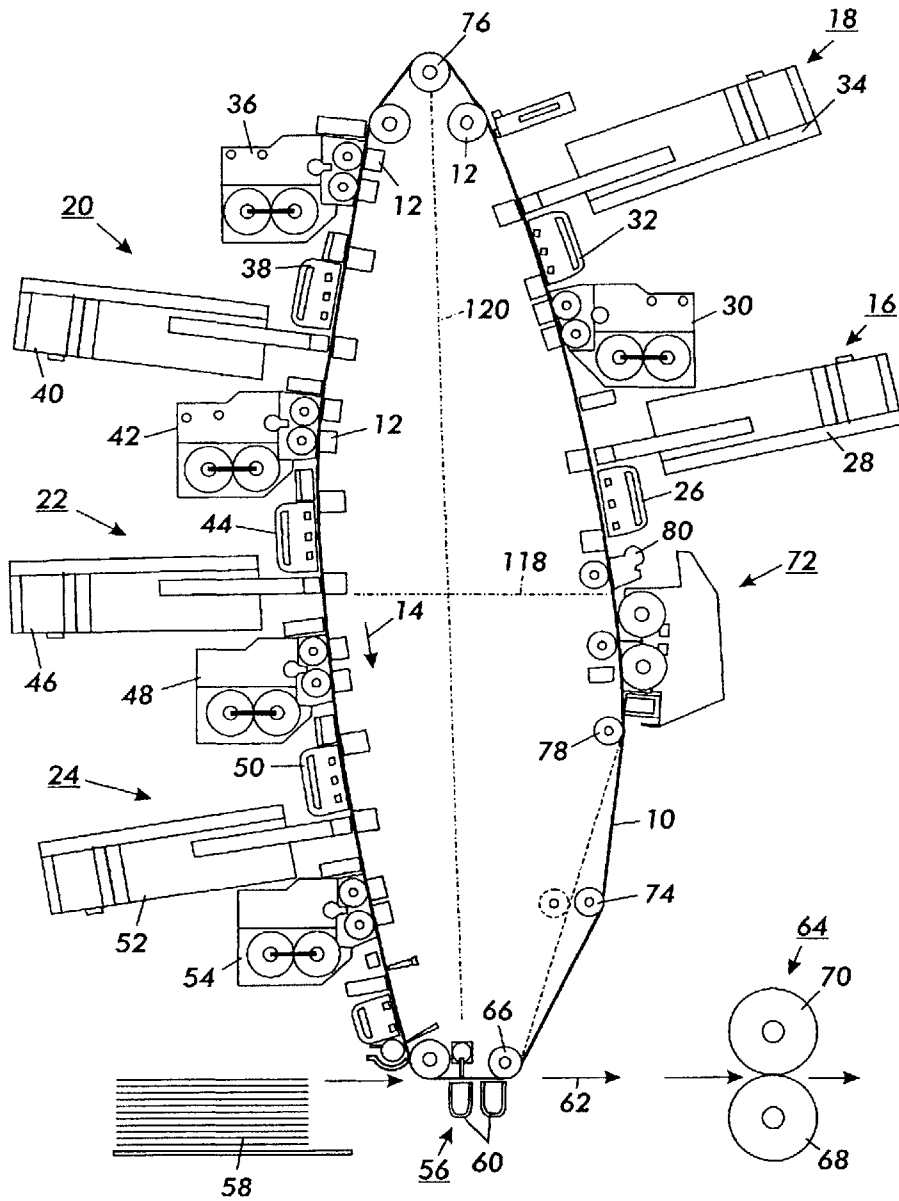


Figure 13: Single-pass Multi-color Architecture [23]

#### 4.4 GROWTH IN COMPLEXITY

As the capabilities of xerographic systems increased to enable the creation of color images, the complexity of the systems also increased. The monochrome systems have a relatively simple architecture. As the number of components increased to enable the desired capabilities, the number of interactions also

increased and the architecture of the systems become more complex. As can be seen from the comparison of the two architectures, as shown in Figure 14, the number of subsystems located around the photoreceptor has increased dramatically. Also, note that the density of the components has also increased. In addition, new interactions determine the location and size of certain components that are not obvious from the illustration. These will relationships will be discussed in the following sections.

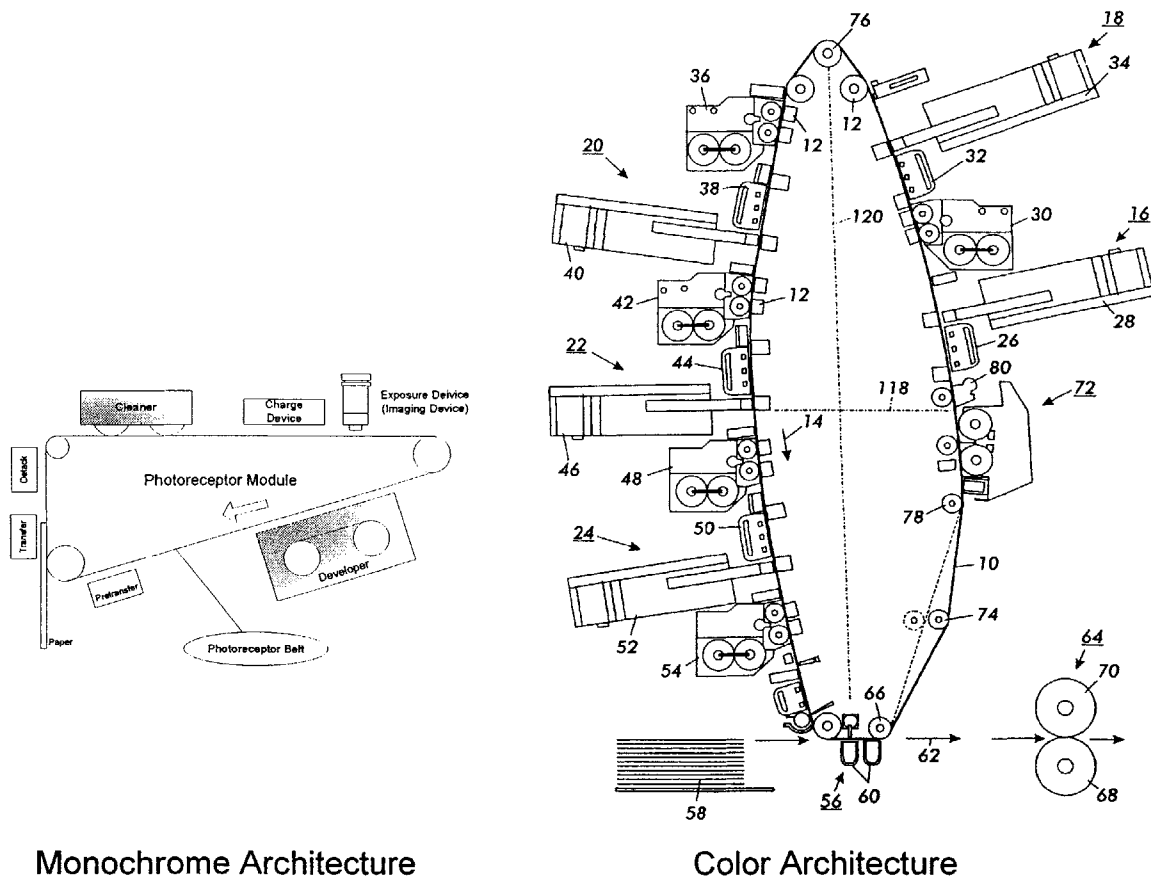


Figure 14: Comparison of Architectures

Originally, it was possible to perform the required architectural analysis with a few simple calculations. The geometric constraints were not significant drivers in

the systems. As the systems became more complex, the analysis became more elaborate with the need to link functional constraints with geometric constraints in a heterogeneous analysis.

#### 4.5 CASE STUDY

In this section of the report the system architecture model for the single pass, multi-color system described earlier will be developed. The case follows the methodology outlined in section 3.

##### DEFINING THE BOUNDARY OF THE MODEL

In attempting to decide the bounds of the architecture model, several possibilities were considered:

- Option 1. The complete xerographic module including all xerographic subsystems and structural supports.
- Option 2. A single xerographic station
- Option 3. The area surrounding the p/r module including the important functional and physical aspects of each xerographic subsystem.

Option 1, the development of the full xerographic module, was determined to be too complex and too wide of scope. The level of effort required to develop a model at this level of detail would be overwhelming. Completing this analysis would be roughly equivalent to completing the detailed module design. Since the arrangement of xerographic elements around the p/r module would have to be determined prior to completing other areas of the module architecture, option 3 would need to be completed as an integral part of option 1. Considering the additional complexity embedded in option 1 would result in delaying the completion of the core architecture decisions.

On the other hand, option 2, a model of a single xerographic station, was thought to be too narrow of a scope. Considering a single xerographic station would not capture the interactions between the stations or system level issues.

Option 3 has the most suitable boundary for this architectural design analysis. The arrangement of the xerographic elements around the photoreceptor module was felt to be the core of the architecture. The models would be used to describe the relationships within a narrow boundary, approximately one to two inches, on either side of the photoreceptor belt. This boundary definition was termed a “waterfront analysis” and is shown graphically in Figure 15.

This selection of the boundary accomplishes the following key aspects of architectural design:

1. The boundary includes all of the important functional interactions required for color xerography. The analysis space includes the xerographic subsystems most responsible for system performance. The desired function is captured within the boundary.
2. The boundary includes the important physical components and their relationships. As mentioned previously, one of the architectural concerns for development of the single pass, multi-color system is the management of the allocation of space around the photoreceptor module. The form is captured within the boundary.
3. Maintains the right level of abstraction. The boundary does not include aspects that could be determined at the next level of implementation detail.

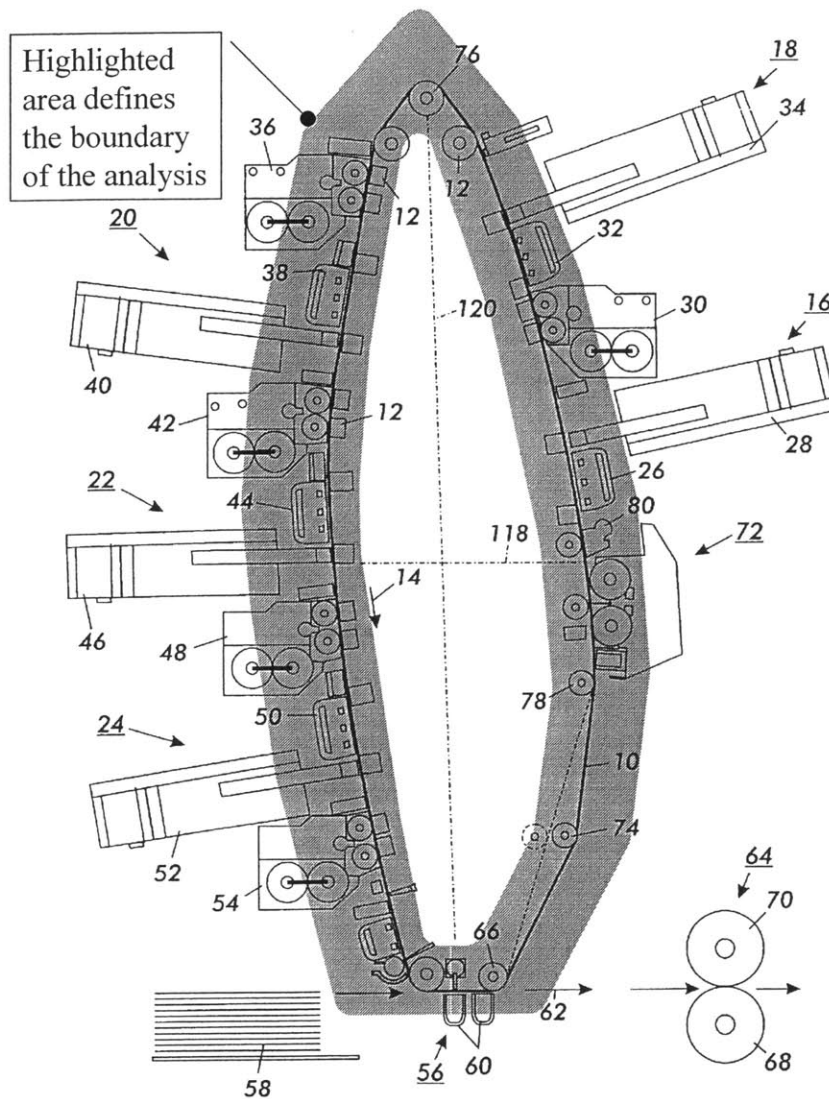


Figure 15: Definition of Boundary (modified from [23])

The key design related outputs from the model are:

1. Specific arrangement and position of xerographic subsystems around the periphery of the photoreceptor module.
2. Specific size and shape of the photoreceptor module. The output from the constraint-based model was used as a template for the production design.
3. Length of the photoreceptor belt
4. Process velocity for the system

It was felt that these four outputs would describe the xerographic architecture in sufficient detail to enable the design teams to continue to the next phase of the design process.

MAPPING OF THE MODEL SPACE.

As stated earlier, one of the decisions that needed to be made early in the process is the scope of the architectural models. Should they include all aspects and all details of the system to create an all powerful system architecture model or should the model be constructed with carefully defined interfaces to other system and subsystem models? The distributed subsystem with an integrative, system architecture, model structure discussed earlier in this paper was adopted. This decision was influenced by the product development and technology readiness processes used by the teams. As part of the technology development process, numerous functional models are developed to aid the development of the subsystem technologies. These models are transitioned to the product development team as an integral part of technology delivery. Therefore, the distributed model approach is the most logical course of development. Figure 16 represents some of the more important models.



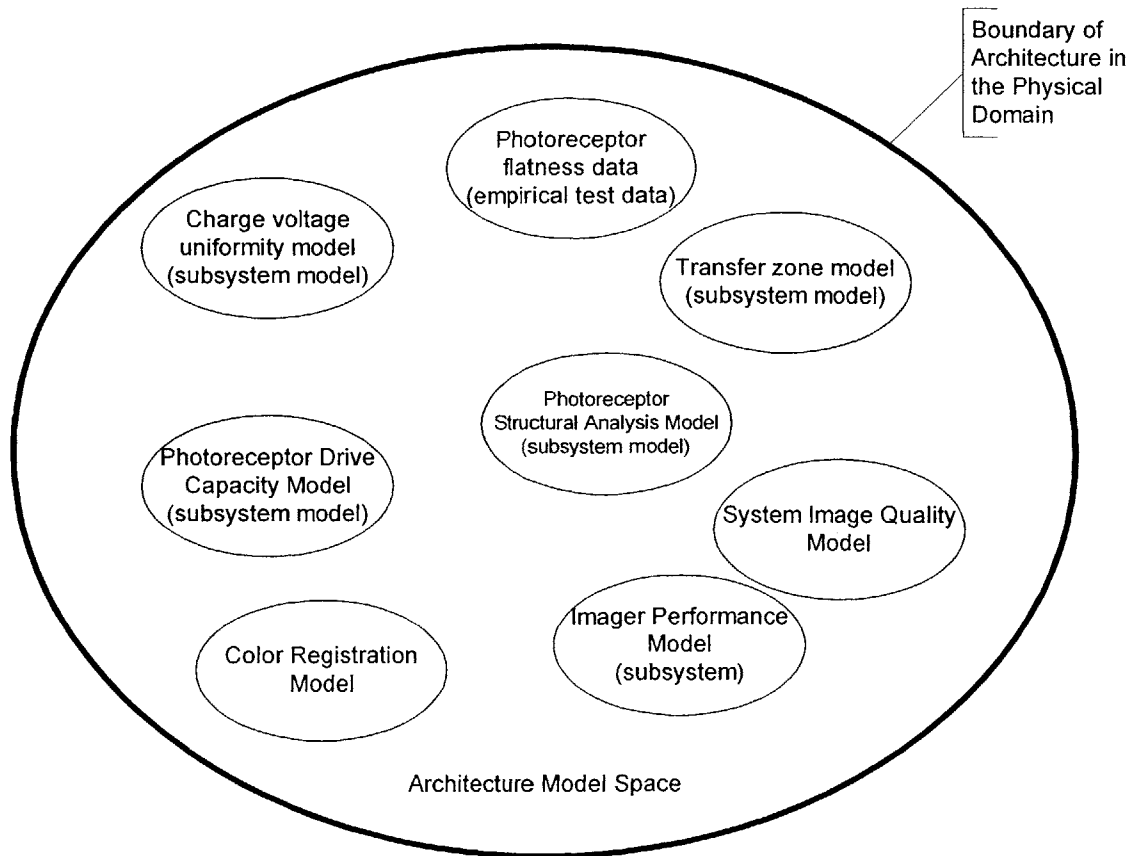


Figure 16: Mapping of Model Space

#### 4.6 PRODUCT REQUIREMENTS, DESIGNS RULES, AND GEOMETRIC CONSTRAINTS

Architectural constraints were collected from numerous sources with the most important being marketing, systems engineering, subsystem/technology teams, and manufacturing. The constraints can be classified into product requirements, system integration rules, and subsystem design rules. Examples of each of these will be discussed below.

##### Product Requirements

- System productivity per unit time
- Maximum system height
- Maximum number of base colors (ie: the number of discrete xerographic stations)

## System Integration Rules

System integration rules are inter-component requirements. They specify a requirement or a constraint between components or groups of components.

Some examples are:

- System Timing
  - The time between end of charge and start of imaging must be greater than or equal to the minimum charge to exposure timing. The discrete value is determined by the formulation of the photoreceptor.
  - The time between end imaging and start of development must be greater than or equal to the minimum exposure to development time. Again, the discrete value is determined by the formulation of the photoreceptor.
  - The time between end of charge and start of development must be less than or equal to the maximum charge to development time. This is a function of the physical sizes of the charge, exposure, and development subsystems as well as the formulation of the photoreceptor.
- Color Registration
  - The image pitch length, the distance along the belt from one imager to the next, should be equal to an integer multiple of the circumference of every rotating member that moves and supports the belt [4].

## Subsystem/Technology Design Rules

Subsystem technology design rules are explicit constraints placed on the design by the chosen technology or are general rules of thumb that have been used in similar design embodiments. The distinction between system integration rules and subsystem technology design rules is somewhat ill defined. In general, a constraint was considered a system integration rule if more than one subsystem was required to satisfy the constraint or if the constraint specified a relationship between subsystems. A constraint was considered a subsystem rule

if the constraint could be satisfied within the subsystem or by the orientation of the subsystem. Some examples are:

- ☑ Developer
  - ☑ Developer augers must be in a horizontal position
  - ☑ Developer to photoreceptor interface scales with process speed in discrete steps.
- ☑ Charge
  - ☑ Length of the charge device is a function of the charge technology and the system process velocity. Scaling is based on discrete steps at which another charge element is added to the charge array.

#### Manufacturability and Serviceability Concerns

- ☑ Spacing between components must adhere to standard serviceability recommendations. For example, there must be at least 5mm clearance between subsystem assemblies that will slide passed each other for service actions.

The list above provides a cross sectional sample of the constraints that needed to be satisfied to develop a suitable architecture.

#### 4.7 A CLARIFYING EXAMPLE

The development of the xerographic architecture quickly grows in terms of complexity and the full disclosure of the details would overwhelm the discussion of the methodology. In order to focus on the methodology while at the same time showing its use in an actual application, this section will discuss a simplified but meaningful example using the design rules and constraints discussed in the previous section. Upon completion of the example, the full system model will be discussed in order to draw conclusions from the actual case study.

#### EXAMPLE DESCRIPTION

The proposed methodology will be used to determine the image pitch length for the system. The image pitch length is the distance along the belt from one imager to the next. It is one of the key-characteristics for the color registration

subsystem and the system architecture. The image pitch length is determined by the physical size of the charge, exposure, and development components as well as the xerographic timing rules that determine the components relative placements. All of these elements have a dependency on the process velocity. Because image pitch spans between the xerographic stations, it is one of the most important considerations in the management of the system's waterfront - the positioning of elements around the periphery of the photoreceptor module. The section discussing the full constraint-based model will provide further discourse on the importance of the waterfront analogy. Figure 17 illustrates the concept of the image pitch length.

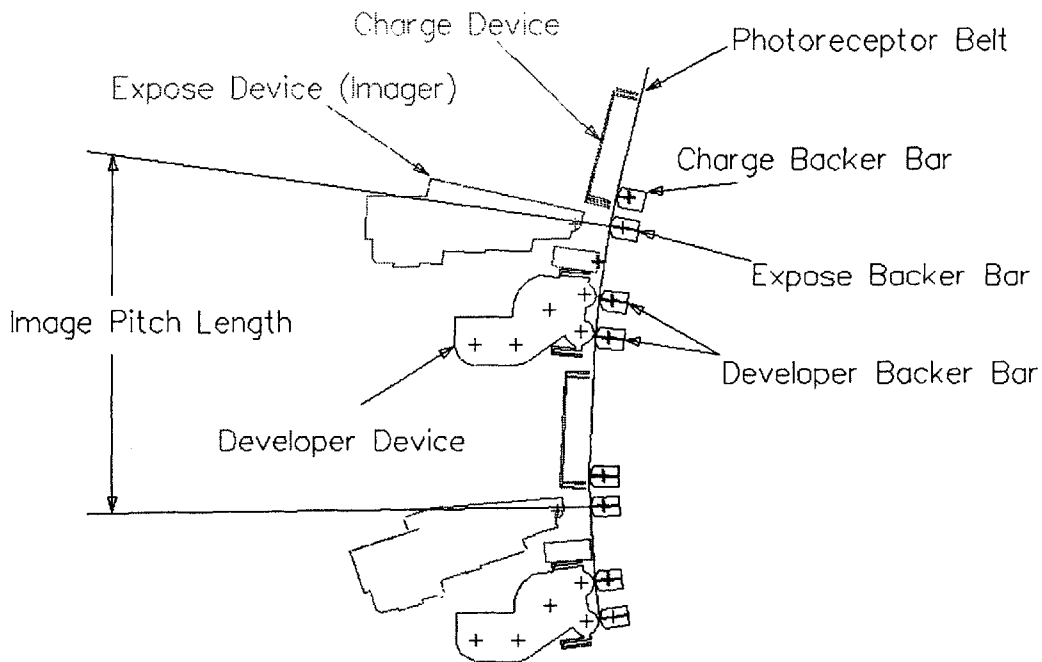


Figure 17: Image Pitch Length Diagram

### IMAGE PITCH LENGTH DSM

Figure 18 shows the partitioned DSM for determining the image pitch length.

The DSM shows the relationships between the image pitch length and the xerographic components, xerographic timing rules, and technology requirements. For example, the DSM shows that the number of developer rolls, length of the charge device, and length of the exposure device (imager), is a function of the process velocity. The developer device length is a function of the number of developer rolls and the process velocity. This information is obtained by reading across the rows of the DSM for each variable. The last row of the DSM shows that the image pitch length is the summation of the physical segments of the photoreceptor belt that are located between each imager such as the spans between the backer bars and the wrap length on the backer bar ( $\text{wrap angle} \times \text{radius}$ ). The spans are dependent on the subsystem lengths, and serviceability clearances or xerographic timing rules. Consider the Charge Backer to Expose Span. The relationships in the DSM indicate that this span is a function of Charge to Expose Distance, Charge Device Length, and Expose Device Length. The DSM also contains the dependencies for these elements. The Charge Device Length and the Expose Device Length are dependent on the Process Velocity. The Charge To Expose Distance is a function of the Charge to Expose Min Time, the Process Velocity, and the Serviceability Clearance. Thus by reading the partitioned DSM, one can extract the hierarchy of relationships that are needed to calculate the image pitch length.

Using these relationships and the geometric representations for the subsystems, a constraint-based model can be created to calculate the image pitch length.

Items	Print Rate (Prod. Req.)	Charge To Expose Min Time (Tech)	Expose To Dev Min Time (Tech)	Servicability Clearance (Serv/Mfg)	Process Velocity	Backer Bar Radius	Expose Backer Bar Wrap Angle	Charge Backer Bar Wrap Angle	Dev Backer Bar Wrap Angle	Expose to Dev Distance	Charge To Expose Distance	Number Dev Rolls	Expose Device Length	Charge Device Length	Dev Device Length	Charge Backer To Exposure Span	Dev Hsg Span	Expose Backer To Dev Backer Spa	Dev Backer To Charge Backer Span	Image Pitch Length	
Print Rate (Prod. Req.)	■																				
Charge To Expose Min Time (Tech)		■																			
Expose To Dev Min Time (Tech)			■																		
Servicability Clearance (Serv/Mfg)				■																	
Process Velocity					■																
Backer Bar Radius						■															
Expose Backer Bar Wrap Angle							■														
Charge Backer Bar Wrap Angle								■													
Dev Backer Bar Wrap Angle									■												
Expose to Dev Distance			0	0	0					■											
Charge To Expose Distance		0		0	0						■										
Number Dev Rolls												■									
Expose Device Length													■								
Charge Device Length														■							
Dev Device Length															■						
Charge Backer To Exposure Span											0		0	0		■					
Dev Hsg Span												0			0		■				
Expose Backer To Dev Backer Span										0					0			■			
Dev Backer To Charge Backer Span				0										0	0				■		
Image Pitch Length						0	0	0	0			0				0	0	0	0	0	■

Figure 18: Image Pitch Length DSM

IMAGE PITCH LENGTH CONSTRAINT-BASED MODEL

The first step is translating the relationships in the DSM to algebraic equations. This is a significant challenge and requires cooperation of subsystem and technology development teams. It is much easier to develop the DSM and the detailed equations in parallel as a series of stepwise refinements so that the detailed representation can be captured at the same time that the dependencies are determined. Figure 19 shows the geometric elements that are required to determine the image pitch length.

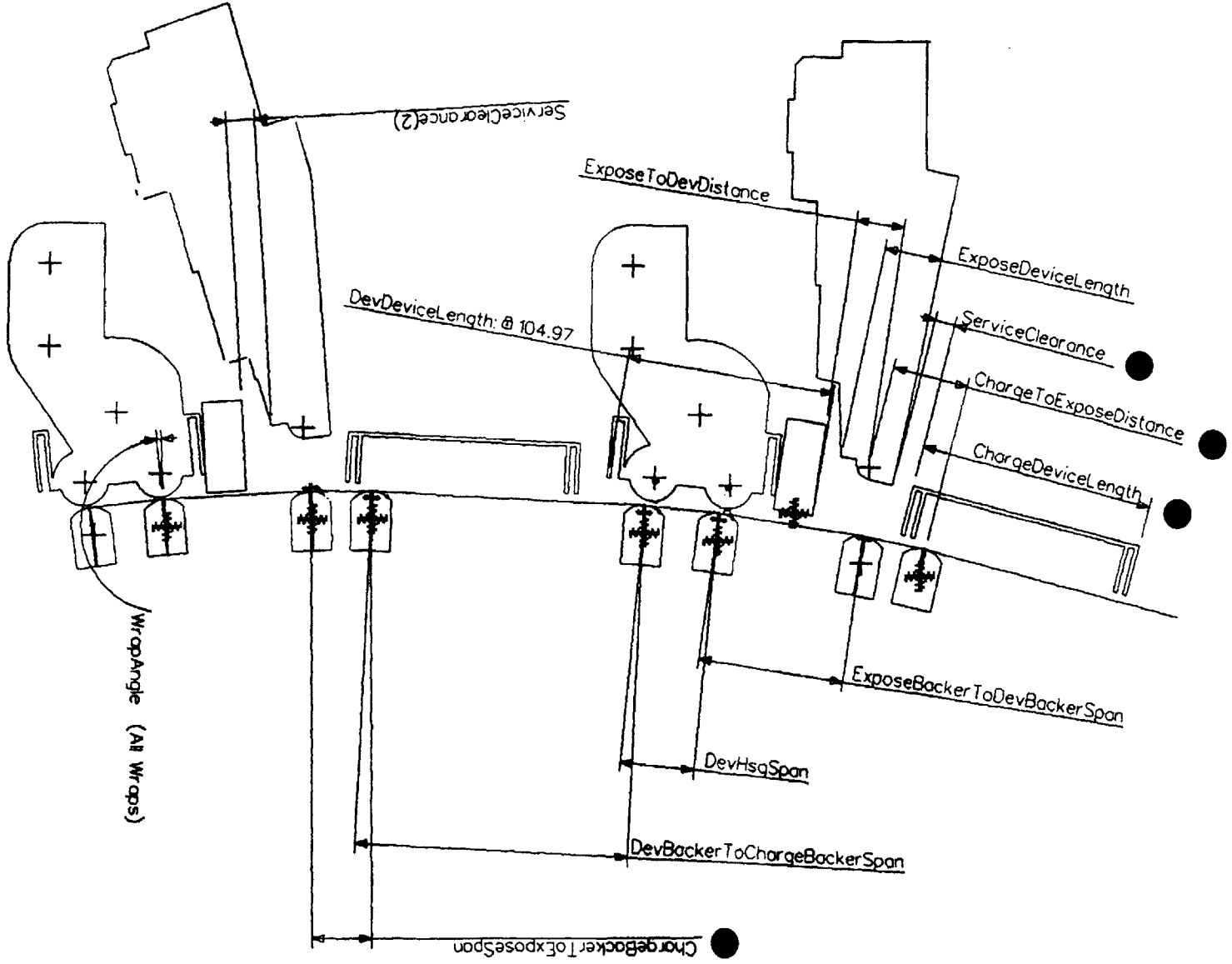


Figure 19: Image Pitch Relevant Dimensions

Within the constraint-based model the dimensions are linked to the algebraic relationships developed from the constraints. The some of these constraints are reviewed below. In order to show the linkage between equations and geometry the variables corresponding to the dimensions are underlined in the following equations. In addition, the dimensions are indicated with a large dot in Figure 19. It is important to note that these are the relationships represented in the DSM. Not all of the constraints are described in the following section, instead an attempt was made to provide sufficient level of detail such that fundamental steps of linking the DSM information to the constraint-based could be understood. For the Charge Device Length, the algebraic equation describing the relationship from the DSM is:

<p><u>Charge Device Length</u> Length of the charge device is a function of the charge technology and the system process velocity.</p> <p>ChargeDeviceLength = ActiveDeviceLength + 2 × AirHandlingDeviceLength ActiveDeviceLength = 2 × DCDeviceMountLength + (NumberOfPinArrays - 1) × PinToPinSpacing</p> <p>NumberOfPinArrays ≥ <math>\left\lceil \frac{V_{Process}}{10.0} \right\rceil + 1</math></p>
--

The Charge To Expose Distance is given by:

<p><u>Charge To Expose Distance</u> Distance is dependent on the charge to expose min time and the serviceability requirements,</p> $\underline{\text{Charge To Expose Distance}} = \text{MAX} \left( \begin{array}{l} \frac{\text{Charge To Expose Min Time}}{V_{Process}} \text{ or} \\ \text{Service ClearanceMin} + \\ \text{ChargeDeviceInternalRelationship} + \\ \text{ExposeDeviceInternalRelationship} \end{array} \right)$
--

Note that the equation contains several geometric parameters whose dimensions were omitted from the figure for the sake of clarity. They are included in the DSM under the Charge Device Length and Expose Device Length variables.



The Charge To Backer Expose Span is given by the following equation.

Charge Backer To Expose Span

This is the length of the span of the belt between the charge backer bar and the expose (image) backer bar.

$$\text{Charge Backer To Expose Span} = \text{Charge To Expose Distance} - \text{ExposeDeviceInternalRelationship} + \text{ChargeDeviceInternalRelationship}$$

Note the equation contains several geometric parameters whose dimensions were omitted from the figure for the sake of clarity. They are represented captured in the DSM as part of the Expose Device Length and Charge Device Length

The Image Pitch Length is then calculated by summing the appropriate spans and wrap lengths of the photoreceptor belt. Again, these relationships are reflected in the DSM.

Image Pitch Length

The image pitch length is the distance between sequential imager stations.

$$\begin{aligned} \text{ImagePitchLength} = & \text{ExposureBackerToDevBackerSpan} + \\ & + (N_{\text{Dev}} - 1) \times \text{DevHsgSpan} + \text{DevBackerToChargeBackerSpan} + \text{ChargeBackerToExposureSpan} \\ & \left( \frac{N_{\text{Dev}} \times \text{DeveBackerWrapAngle} + \text{ChargeBackerWrapAngle} + \text{ExposureBackerWrapAngle}}{\text{ChargeBackerWrapAngle} + \text{ExposureBackerWrapAngle}} \right) \times \text{BackerBarRadius} \end{aligned}$$

The example presented here showed how the DSM and constraint-based modeling work together as integral parts of the methodology. The DSM is used to map the relationships. The detailed algebraic constraints for the constraint-based model are developed through successive refinement of the dependency relationship. The DSM provides a graphic mapping of the relationships that are represented in the constraint-based model. The DSM can be used as a guide in the development of the more detailed model. The DSM highlights the dependencies without providing an overwhelming amount of detail. While the constraint-based model captures the level of detail required to perform the actual

calculations. The DSM and the constraint-based model have complementary attributes that help to form a complete methodology and enable discussions about the system on multiple levels.

#### 4.8 XEROGRAPHIC ARCHITECTURE MODEL

The remainder of this paper will discuss the use of the methodology on the development of xerographic module architecture and draw conclusions on its applicability.

#### CONSTRUCTION OF SYSTEM DSM

The detailed constraint-based model contained over 140 variables. A DSM using this level of detail would be 140X140. Although not a tremendously large matrix in terms of linear algebra, it is very large in terms of the DSM's primary function within this methodology. The purpose of introducing the DSM technique into the methodology was to communicate the dependencies in the architecture in a clear and succinct manner. The intention is to provide a document that describes the architectural relationships within a page or two. Clearly, a matrix of 140 rows and columns would not be considered succinct. In addition, it is hard to imagine that such a document would be useful to the design teams. Therefore, the most important consideration in developing the DSM is the selection of the appropriate level of abstraction.

Choosing the level of abstraction for the DSM aggregates portions of the architecture to reduce the complexity of the resulting DSM matrix. This aggregation is a clustering of the architectural relationships before creating the DSM and must be done with care.

One possible approach is to aggregate the architectural elements to be consistent with the distributed subsystem with an integrative system architecture, model structure adopted earlier. Internal subsystem relationships are aggregated into a single element in the matrix. For example, the charge subsystem size is an aggregation of three or four variables that determine the size. System level or integrative relationships are kept explicit. Thus, the recommended elements consist of aggregated intra-component relationships and explicit inter-component relationships. This approach maintains the intended function of the DSM technique by reducing the complexity of the resultant matrix and improving its readability.

#### DSM REVIEW

Figure 20 shows the DSM matrix developed for this model. Several iterations were required to improve the readability of the matrix. In addition some details have been omitted due to their proprietary nature. Further modifications could be made to further reduce the complexity and improve the readability. The DSM shown in the figure was built using Steward's PSM32 code. The DSM has been partitioned and one tearing operation was performed to help to simplify the large iteration loop in the lower right of the matrix. Tearing is a DSM technique in which a set of feedback marks are chosen such that if removed from the matrix repartitioning will render the matrix lower triangular. The marks that are removed from the matrix are called "tears". Tears can be thought of as an initial guess. For this DSM structure, the belt length was selected as the tearing element. This selection was based on the assumption that knowledge of the belt length of the technology development fixtures would provide an adequate initial guess. This

also reflects the development of the constraint-based model, which required the belt to be a closed loop, and therefore a known quantity before the other constraints could be satisfied.

The DSM shown in the figure has three iteration loops. The first deals with the wrap angles on the various rolls and support bars (called backers or backer bars). The upper and lower triangles are nearly full which implies that its interactions are tightly coupled. This is because the number of degrees of wrap is limited to 360 degrees. With numerous elements requiring belt wrap due to subsystem and technology constraints there is contention in satisfying the constraints. Numerous iterations are needed to resolve the constraint network. The majority of this loop is captured in the geometry elements in the sketch note of the Mechanical Advantage application.

The next two loops need to be considered together for one of the loops is embedded in the other. First consider the larger loop. This loop deals with determining the layout of the arrangement of components around the belt module, calculating the image pitch length, the belt length, and process speed for the system. The belt length can be thought of as the summation of the image pitch lengths, the roll wrap length, the cleaner and tension mechanism spans, and the remaining spans needed to “wrap” the belt around the belt module.

The purpose of the large loop is to determine the size of the image pitch length, the size of the rolls, the resultant belt length, and the process speed required to meet the product print rate requirements.

The inner loop determines the placement of the cleaner and preclean erase with respect to the first charge device. This placement is determined by the size of the cleaner device as well as a xerographic timing rule.

These two loops are captured as algebraic equations in the Math Note of the constraint-based model and are linked to the appropriate geometric elements in the Sketch Note in the manner described by the example shown in Figure 9.

The DSM does a good job at highlighting the interactions in the development of the architecture. It shows the areas that are tightly coupled and captures the important system and subsystem level constraints that influence the architecture. Although it contains nearly 80 elements, the matrix can be used to as a communication tool. The important aspects of the architecture and the critical dependencies are clearly highlighted.

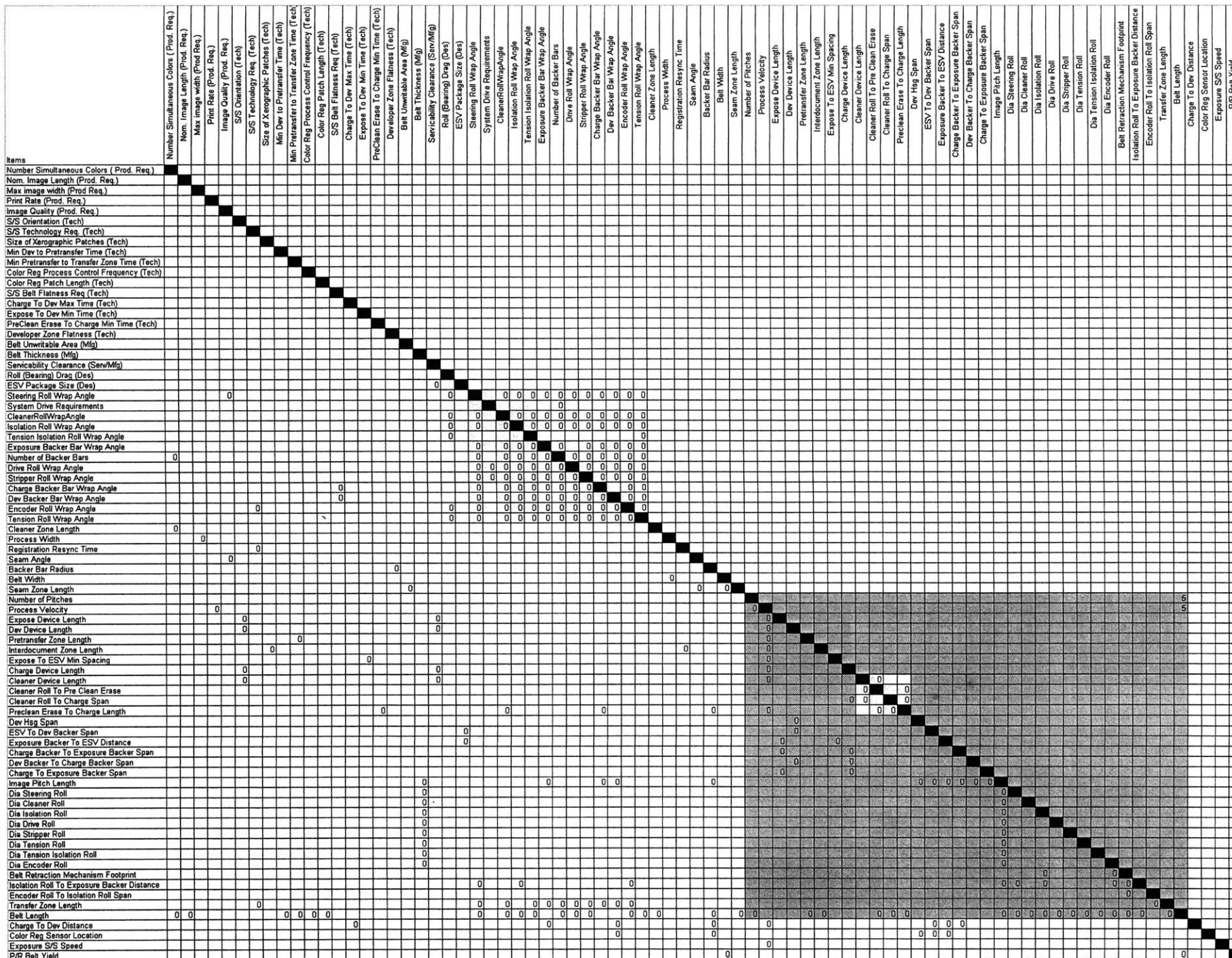


Figure 20: Architecture DSM

Although the DSM was not used during the development of the actual architecture, the Mechanical Advantage application has similar capabilities on a variable by variable basis.

One of the features of the Mechanical Advantage environment is called "Show Connected Items". When selected this feature provides dependency information for the chosen variable by highlighting the appropriate variables in the environment windows. The option window for this feature is shown in Figure 21. The first option is "Show what drives me directly". This option is equivalent to reading across the row of a DSM element. "Show what I affect directly " is equivalent to reading down the column of a DSM element. The options titled "Show what drives me at all" and "Show what I affect at all" highlight direct and indirect dependencies.

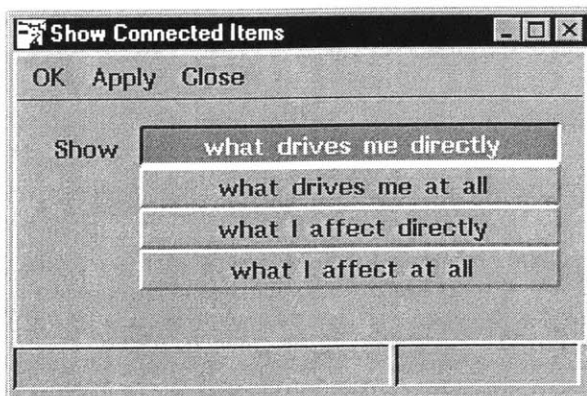


Figure 21: Show Connected Items

This capability was used during the development of the architecture model to explain the dependencies. In fact, this capability was one of the primary reasons the DSM technique was included in the methodology. The DSM communicates

the same level information for all the variables by using the matrix structure described in previous sections. Use of the DSM technique enabled better communication by presenting a "systems view" of the architectural dependencies. Thus although the original effort did not use the DSM technique, it did have access to many of the benefits of the DSM technique.

In order to quantify the value of the DSM technique when used in combination with a constraint-based model, several members of the design teams that participated on the architecture development task were asked to review the DSM matrix. The lead designer for the photoreceptor module provided the following comments.

"...[T]he DSM looks like a nice supplement to the [Mechanical Advantage] model. I like that it shows the dependencies so clearly compared to ... seeing the dependencies in the [Mechanical Advantage] model alone. I'm not sure you could locate and visualize all of these relationships from the [Mechanical Advantage] model without generating some other document to display it. The DSM is graphical, easy to read, would be a nice communication tool, would assist in interface discussions, and would assist in creating a system. When I first saw this DSM format, it reminded me of [a] QFD model. That model showed all of the relationships between the System level Critical Parameters and the Critical Specifications that flowed into them. Why couldn't this be used to help create a software architecture or any relatively complex system with a lot of relationships?"

The attributes mentioned - easy to read due to its graphical nature and a good communication tool - are the reason that the DSM technique was selected as the "front end" of the methodology. The next section discusses the development of the constraint-based model.

#### CONSTRAINT-BASED MODEL CONSTRUCTION

As mentioned earlier in this paper, the boundary definition for the model was selected such the constraint-based model describes the relationships within a narrow boundary, approximately one to two inches, on either side of the photoreceptor belt. This boundary definition was termed a "waterfront analysis"



and is shown graphically in Figure 15. The physical elements of the waterfront are the xerographic subsystems and the backer bar support members. Backer bars are used to support the photoreceptor. Since the photoreceptor is wrapped around the backer bars, they responsible for providing the shape of the architecture. Figure 22 shows the waterfront elements for two xerographic stations.

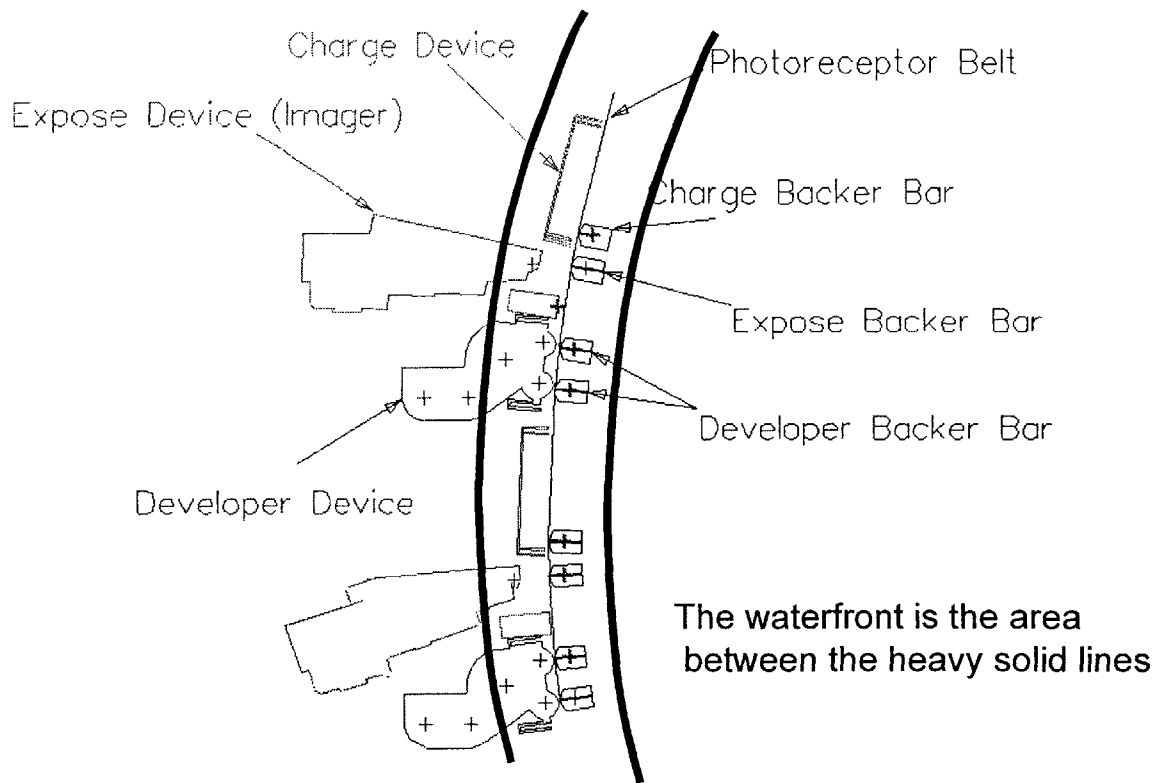


Figure 22: Waterfront Illustration

The waterfront analysis is appropriate for the analysis because it includes the required functions as well as the required physical components. In addition, framing the problem in terms of a waterfront analysis constrains the geometric

domain of the problem to two dimensions instead of three. This significantly reduces the complexity of the problem. The terminology of waterfront analysis was chosen because the resultant architectural layout looks like a “map” of the area surrounding a body of water, i.e. picture a map showing the buildings surrounding Boston Harbor. This selection is further supported by the fact that the xerographic portion of the print engine is primarily a planar machine and therefore amenable to two-dimensional analysis.

The few constraints acting in the third dimension that affect the architecture, such as the seam angle and maximum desired image width, were handled by including their influence in the algebraic constraints.

The creation of the constraint-based model consisted of three primary activities.

1. Creation of constrained geometric objects to represent the important features of the xerographic subsystems.
2. Translation of the constraints, product requirements, subsystem design rules, technology requirements, and system integration rules, into algebraic and logical equations.
3. Building the integrated assembly model such that geometric components are properly constrained with geometric and equation based constraints.

The development of the geometric objects required close work with the subsystem teams. In many cases the CAD geometry from the technology prototype was imported into the Sketch Note and then modified, constrained, and integrated into the system model. The original versions of the model included the full geometry for all the subsystems. One drawback of this approach was that the process had to be repeated every time there was a significant change in the subsystem design approach. This tended to slow the development process.

Eventually a better approach was developed in which the critical waterfront dimension was identified for each subsystem and incorporated into the model. Updating a single dimension was significantly quicker than rebuilding a portion of the model.

Translating the constraints into equation sets posed a significant challenge and required the cooperation of the subsystem and technology development teams. This process was an iterative process requiring close communication with the technology owner.

The creation of the constraint-based model used the same progression as shown the Image Pitch Length example.

#### CONSTRAINT-BASED MODEL RESULTS

The model was developed in the Mechanical Advantage environment using the process described in the previous section. The model was able to produce the four design outputs required of the architecture development phase. These outputs were:

1. Specific arrangement and position of xerographic subsystems around the periphery of the photoreceptor module.
2. Specific size and shape of the photoreceptor module. The output from the constraint-based model was used as a template for the production design.
3. Length of the photoreceptor belt, which enabled an early estimate of the manufacturing yield and field service costs.
4. Process velocity for the system.

The following sections will provide more details and discuss the application of this methodology within the context of the product delivery process.

#### 4.9 ORGANIZATIONAL STRUCTURE AND ROLES AND RESPONSIBILITIES

Before discussing the alignment of the architecture development methodology, one more topic needs to be discussed with respect to developing the architecture model. The topic is the organizational behavior that is required to support the development activity.

##### SENIOR MANAGEMENT INFLUENCE

Support of this activity by the senior design managers on the program was critical. Without their support the time and effort required to develop the models would not have been available. In particular, the Xerographic Design Manager played a pivotal role the genesis of the methodology as well as championing the process and “buying time” so that the methodology and required skill set could be developed.

From the start of the effort to the successful sign-off on the architecture was approximately eighteen months. This is longer than most architecture development activities. Xerographic Design Manager and Mechanical Design Manager were strong supporters of this development process and provided support, resources, and encouragement until the process yielded the results. They felt that they were making the up-front investment to avoid problems during the detailed design phase.

##### TEAM DEVELOPMENT PROCESS

The development of the architecture was truly a team effort and several aspects of the team interactions are worthy of comment.

The development of the initial set of constraints was developed through a series of one-on-one conversations with the technology development teams as

well as the subsystem development teams. These meetings transferred much of the knowledge that the teams had developed.

Once the initial model was developed and producing reasonable results, the team held a weekly architecture meeting. The purpose of the meeting was to share the status of the architecture, and capture issues with respect to the model outputs. This forum brought together the appropriate members from the technology development group and product development team to discuss and resolve disagreements and concerns on the implementation of the dependencies. It was in this forum that agreement was reached on prioritization of conflicting constraints. Senior managers were key participants in the meeting.

Without the organizational behavior from the technology and product development groups, the constraint-based architecture approach would have failed. The model played the role of integrating the teams' combined knowledge to produce a workable architecture.

#### 4.10 ALIGNMENT OF METHODOLOGY WITH PDP PROCESS

One of the important considerations in developing the methodology is to determine how well the methodology is aligned to the product development process. A high value methodology will be closely aligned to the product development process and its outputs will enable the product development team to meet its deliverables with a higher quality product with less effort.

The methodology described herein has been used throughout the product development cycle. As mentioned earlier, this case study covers the system development from the product pre-concept stage to the detailed design stage.

The next section describes the how the architecture models have been used as part of the larger product development process.

#### MULTIPLE USES OF THE MODEL DURING EARLY DEVELOPMENT PHASES

Several architecture models were developed during the early stages of the product development. The models progressed from abstract models that were suitable for making high-level architecture decisions to models that contained enough detail to help guide the detailed design.

#### Select Technology - using high-level model for architectural narrowing

During the pre-concept phase of the development process, basic questions need to be answered about the product. For example, the basic technologies need to be selected as does the high level architecture or system design. At this stage, the methodology can be used to develop high-level architecture models. These models capture the "critical few" aspects that describe the decision the team needs to make.

For the xerographic architecture the questions that needed to be answered were:

- (1) What type of photoreceptor archetype should be used a belt-based design or a drum-based design?
- (2) For the belt-based system, how should the individual xerographic stations be positioned around the photoreceptor module. Should all the xerographic stations be on one side of the module or should stations be on both sides of the module?

The architecture development team decided to use a formal decision process known as AHP (Analytical Hierarchy Process) to make the decision on the type of photoreceptor system to use. The AHP decision model compared three high level architectures that included a drum-based system and two belt-based

systems. Architecture models were developed which provided estimates of size, belt (drum) length, and process speed as well as an approximate layout for these systems. The constraint-based models used to describe these candidate architectures were kept as simple as possible. Rules of thumb and best guess estimates were used in the place of detailed design rules. For example, an approximation of the image pitch length based on estimates of the subsystem device lengths was used to construct the models. Calculations were kept to a minimum. The intention was to understand the shape and size of the various architectures. Constraint-based models were used to help to ensure that the same "rules" were applied to each architecture. This enabled the architectures to be compared on an equal basis. This was important because during the early stages of the project many of the design rules and technology guidelines were still being developed and were changing rapidly. It was essential to make sure that the candidate architectures were developed using the same set of rules. Without this, the basis for comparison would be lost. This information was used as inputs to the team's decision model. Figure 23 in the next section shows the layout of these architectures.

Upon the program's decision to use one of the belt-based systems. A more refined model was developed that enabled the team to make the decision on the orientation of the xerographic module and a further refinement of the technology selection. The important area of concern for this decision was the height of the transfer zone and access to the zone for jam clearance. The architecture models indicated that the module height would be such that the total system would be

over six feet tall. If the transfer zone was placed at the top of the module, then the paper path geometry and jam clearance requirements became much more difficult to achieve. On the other hand locating the transfer zone at the bottom of the module would allow more reliable paper path geometry and provide easy access for jam clearance, but would be a major change from previous products and development fixtures using similar technology. This decision was made based on the architecture model's ability to estimate the module height to a high level of accuracy. Plywood models of the system were developed based on the output of the model. Based on this information the program selected the bottom transfer option.

*Concept Phase - using detailed architecture model*

The next stage of the architecture model development was to create a detailed model of the system using the full set of constraints and requirements. This model was refined until all product requirements were met. One of the outputs from the model was a two-dimensional CAD file of the waterfront area. In particular the product development teams needed to know the shape of the photoreceptor module. This was provided by creating a two-dimensional CAD file of the photoreceptor belt. This file was imported into the CAD system and served as the template for the module design. As the model was refined and improved, the CAD template was updated to reflect the latest results. This process helped to coordinate the system design. This step of the process helped to close the loop with the design teams. As mentioned earlier in the paper, when the design teams made a modification to their subsystem design, the changes were imported from the CAD system into the constraint-based architecture



model. The loop was closed when the revised output from the architecture model was imported into the CAD system to serve as the next version of the architecture template.

#### *Design Phase - using the detailed architecture model*

During the detailed design phase of the project, the architecture model served a different purpose. As the design progressed and the level of detail increased the architecture model ceased to be the center of the development activity. However, it still played an important role during this stage of development. The role is best illustrated by an example.

During the detailed design stage, testing from some of the xerographic test rigs indicated that an additional charge device was required for the first xerographic station. However, that area of the module's "waterfront" was already densely packed with other subsystem components and there was not enough room to add it in. One of the more powerful features of the Mechanical Advantage application is that the user can change the constraint network. Variables that were set by the user can be "unlocked" so that they are calculated by the constraint network and variables that were constrained can be "locked" so that the user can change them. This capability enables the users to reconfigure the model such that only certain types of changes are allowed. This capability enabled the model to be used to develop a "minimum impact" solution for adding the extra charge device. In this case the system level outputs determined during the architecture development phase of the project such as the size of the module, the length of the photoreceptor belt, and the process speed were not allowed to change. All lower level changes had to be made without changing

these values. By modifying the constraint network and manually tracking system level changes, a solution was developed that had minimal system level impact.

In another example, testing indicated that the stripper roll was too large and lightweight paper could not be reliably stripped from the photoreceptor belt. This condition would result in unacceptable reliability problems for print jobs using those paper types. The model was used to modify the waterfront while maintaining the system level variables.

In both of these examples, the architecture model was used to develop a solution to a problem that could have changed the architecture and caused the design to deviate from the system and subsystem rules that constrain the design. The model was used to maintain the integrity of the architecture as the design matured. In this context, the architectural integrity is the satisfaction of the set of constraints that were originally developed with the DSM and implemented in the constraint-based model. Without the use of the constraint-based model to ensure that the design changes did not violate the constraint set, the proposed design modifications could have resulted in a product design that no longer met its intended function. More importantly by ensuring that the system satisfies the constraint set, the product development team can have a higher level of confidence that the detailed design implementation will meet its goals.

In summary, the methodology was used for the following purposes.

PDP Phase	Methodology Usage and Alignment
Pre-concept (Selecting Technology)	Development of high level model for architecture narrowing. Selection between archetypes Transfer zone location (module orientation)
Concept Phase (Define Product)	Detailed model for architecture development and refinement Providing an architecture template for the design task
Design Phase	Update detailed model to reflect new knowledge, technology/design refinements, and requirement changes refinements.

As can be seen from the table, the architecture model developed with the methodology described in this paper is closely aligned with the product development process. In addition, the model continues to add value after the selection of the architecture and the program focus has shifted to the detailed design. During this phase of the process, the model can be used to maintain the integrity of the architecture.

#### 4.11 PROJECT TIMELINE

This section provides a graphical timeline showing the use of the model during the different phases of the product delivery process. During the pre-concept phase, the multiple models were developed to support technology selection. The level of detail in the model was increased to develop more fully the proposed architecture. Over twenty-two versions of the constraint-based model were developed to analyze alternate technology and design implementations. Approximately 18 months into the project the senior management team had enough confidence in the architecture to freeze the architecture. This was a significant decision, requiring the commitment of hundreds of millions of dollars over the life of the program. Since that time the

architecture has not changed. Minor design changes have been made and the model has been used to guide those changes to preserve the integrity of the architecture. Without the use of the constraint-based model to ensure that the design changes did not violate the constraint set, the proposed design modifications could have resulted in a product design that no longer met its intended function. More importantly by ensuring that the system satisfied the constraint set, the product development team had a higher level of confidence that the detailed design implementation would meet its goals. The timeline is shown in Figure 23.

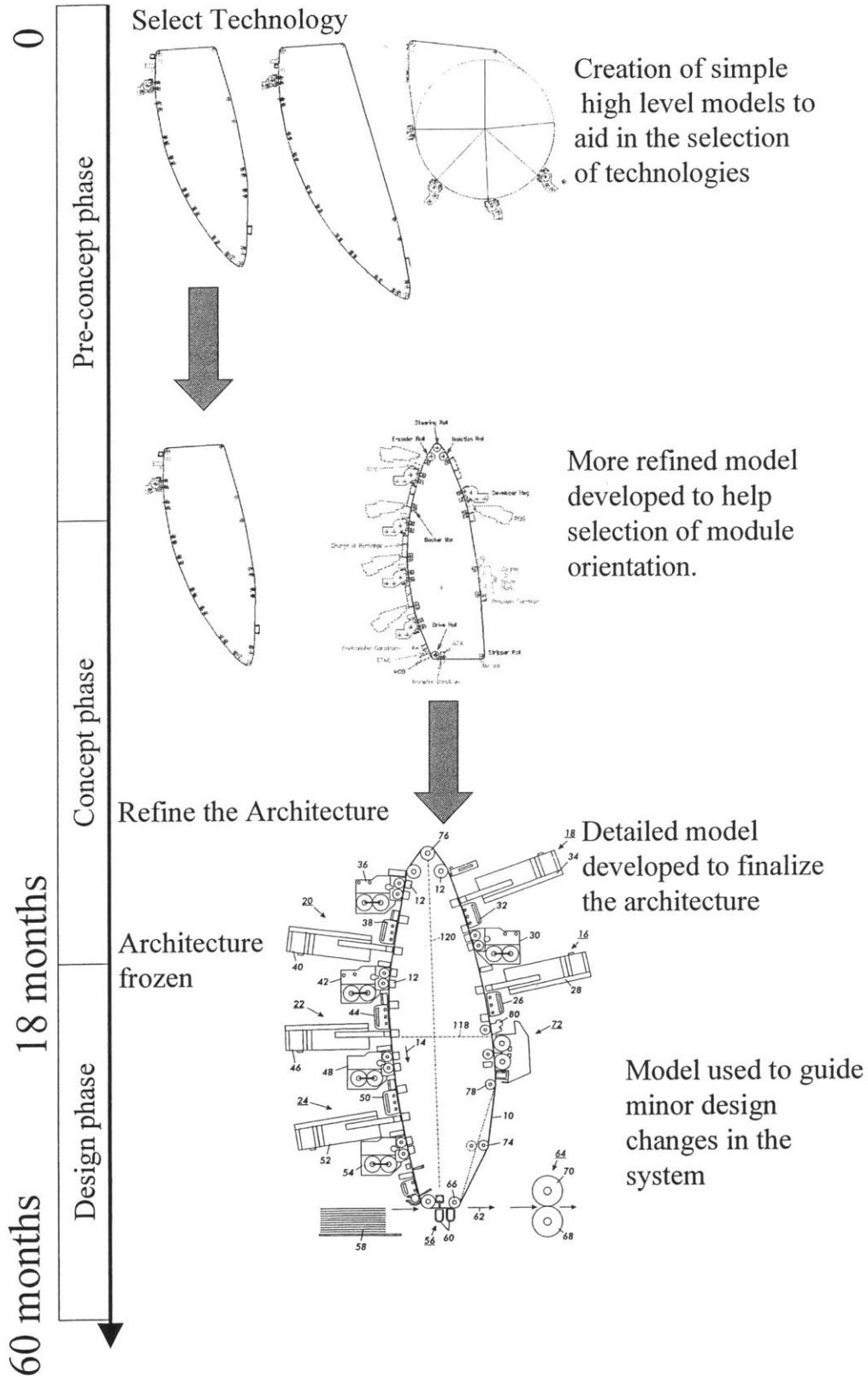


Figure 23: Project Timeline

#### 4.12 LESSONS LEARNED

Based on the development of the methodology and its application to a system development process the following lessons have been learned.

1. The DSM technique and the constraint-based model both can play an important role in the development of system architecture. The DSM is well suited to map the system dependencies. The graphical representation enables visualization of the relationships at a system level. It is well suited to be used as a documentation and communication tool. The constraint-based model provides the analytical capabilities to turn the dependencies into a constraint network and produce detailed description of candidate architectures. The strengths of the two techniques are complementary. When combined with the organizational structure that supports their use the methodology can be successful.
2. The waterfront analysis is the right level of abstraction for this type of system. It captures the right level of detail without including unnecessary design detailed that were best left for a later development phase. The selection of the boundary included all of the xerographic functions, as captured in system integration rules, as well the form of the subsystem components.
3. At the beginning of the paper, it was noted that prior research indicated that the development of a copier required over 1 million decisions [8]. In order for this methodology to be successful it is important to limit the scope of the model so that only the information regarding the "critical few" decisions that affect the architecture are included in the model. The scope of the model is determined by the boundary definition as well as the level of detail included in the model.
4. The architecture model, although important as a technical development tool, is equally as important as an organizational tool. By having the model development as the central focus point for the architectural development activity, organizational behavior is focused. It enables bringing together the right people to review the results of their inputs. Reviewing the model's output on a weekly basis helped to raise system level issues. This served to focus attention and align work priorities. In addition, this organizational structure helped to promote co-production between the technology and product development organizations and helped to improve organizational learning.
5. The methodology is closely aligned to the product development process. During the different phase of the development process, the models play different roles. During the pre-concept and product definition phases, the

model plays a central role in the development of the architecture. During the detailed design phase, continued use of the model helps to maintain the integrity of the architecture as the design matures.

6. The methodology is uniquely suited for a "clean sheet" design project. On many clean sheet projects the level of domain expertise is low because of changes in product scope as well as multiple new technologies, which are not well understood. The methodology helps to capture and expand the domain knowledge. In addition, the form of the technology implementation and therefore the technology driven subsystem-KCs are not understood until the architecture is defined. On a variant product, many of these issues have been resolved and the domain knowledge is very high and well integrated across the organization. Although the methodology could be used on a variant, the returns on the effort will not be as high.

#### 4.13 UNIQUE VALUE CREATED BY SYSTEM/ARCHITECTURE MODEL

The methodology described in this paper leads to the development of a system architecture model consisting of a DSM matrix and a constraint-based model. These models add unique value to the development process. This section will examine some the value adding capabilities of the methodology.

#### *ROBUSTNESS OF THE METHODOLOGY*

The xerographic module is one of the two mechanical modules in the print engine. The other significant module is the paper path module. The paper path architecture was developed using a different methodology. Comparing the number of system level changes to the architecture during the detailed design phase provides one informal measure of the capability of the methodology in that application.

As mentioned earlier, the xerographic module went through several design changes that could have affected the module architecture. In addition to the two examples discussed in the case study, two additional design changes could have affected the module architecture. The impacts of these changes were minimized

by using the models to ensure that the design changes continued to satisfy the architecture constraints. The changes could be incorporated into the next design release and were considered to be minor changes. The paper path architecture experienced a similar maturing process. However, three problems required significant change to the original architecture.

There is no guarantee that the application of this methodology to the paper path would have produced better results. There are obvious limits to what can be captured in a model and emergent problems are difficult to anticipate. However, the methodology provides a good foundation from which to make design tradeoffs. In its one application, it has been very successful, although more cases are needed to understand the extent of its capabilities. Given that this project was the first application of the next generation of color xerographic technology consisting of seven new subsystem technologies, it is significant that the architecture developed, using many of the key attributes of this methodology, has not changed in nearly four years.

#### CREATION OF INTELLECTUAL PROPERTY

Another measure of the value of the methodology is its ability to create intellectual property. Because of the architectural modeling effort, two invention proposals were filed. One has already been issued as a United States Patent [23]. A copy of the patent is included in Appendix A. The other application is still under review.

#### **5 CONCLUSIONS AND SUMMARY**

This paper developed an architecture development methodology by combining the two powerful techniques, the DSM technique and the constraint-



based model, and placing them in an organizational structure that championed their use. A case study was reviewed in which the architecture of a xerographic module for a color printer was created using many aspects of this methodology. The case study made extensive use of the constraint-based model. The DSM technique was added to the methodology based on the perceived shortfalls in ease of communication in using the constraint-based model code to describe the architectural dependencies. The judgement of the usefulness of the DSM was based on the input from important participants in the development activity.

#### VALUE OF DSM TECHNIQUE

The primary decision that needs to be made when constructing the DSM is choosing the right abstraction. By choosing a suitable level of abstraction, the DSM can be made to convey information at the level of detail suitable for the intended audience. The case study, the DSM was developed after the constraint-based model and required several iterations to remove levels of complexity. If the DSM was developed prior to or in parallel with the constraint-based model, the level of detail might be easier to control.

The primary value the DSM technique is that it creates the ability to communicate architectural dependencies without having to work at a high level of detail. In addition, it provides reference documentation that the design teams can use to know when and with whom they need to discuss proposed changes.

Another benefit is that the DSM provides a visual aid in the development of the constraint-based model. Since the DSM can be partitioned to minimize iterations or clustered to group related constraints; the DSM can be used as a guide to structure the constraint-based model. In the case study, the DSM was

partitioned and indicated a possible structuring for the creation of the constraint-based model. This structure was very similar to the structure that was used in the construction of the actual model.

Input from members of the design team supports these conclusions.

#### VALUE OF THE CONSTRAINT-BASED MODEL

The primary value of the constraint-based model is the ability to develop a high quality abstraction of physical design based on known constraints. The Mechanical Advantage application provides an environment in which geometric objects, geometric constraints, and algebraic equations can be combined to create the constraint-based model. This type of model is able to develop a representation of the system architecture that can be used to drive design decisions.

In the case study, the model provided high quality estimates of critical system level variables. It was also used to determine the location of components around the waterfront of the module. The flexibility of the development environment enabled detailed "what-if" analysis of different design rules, technology limitations and integration rules. One of the most useful outputs from the model was the architectural template that was used to guide the detailed design.

It was also shown that the constraint-based model continued to play an essential role in the system development as the program completed the detailed design, by helping to maintain the integrity of the architecture as the design and technologies matured.

When these techniques are coupled with a proactive management team and an organizational structure that supports model-based learning the methodology produces significant and meaningful results.

In section 2 of the paper, four needs that an architecture methodology should satisfy were developed. These needs were:

1. Provide a means for modeling the system that provides the capability to gain feedback on proposed decisions. This promotes rapid system learning.
2. Provide a definition of the linkage between product requirements and design parameters. From a key characteristic point of view, this creates a map from product requirements (Product-KCs) to subsystem parameters (subsystem-KCs).
3. Provide documentation that makes the architecture explicit and enables others to have access to the architectural knowledge (decisions).
4. Increase confidence in the proposed system so that product design can proceed with a minimum of risk.

Based on the details of the case study it is now possible to answer the question of "does the methodology satisfy the four needs of architecture development?"

Requirement 1: Provide a means for modeling the system that provides the capability to gain feedback on proposed decisions. This promotes rapid system learning.

The constraint-based model provides the means for modeling the system.

The organizational structure and behavior surrounding the development of the model promotes organizational learning and creates the environment in which immediate feedback can be received on proposed decisions.

Requirement 2: Provide a definition of the linkage between product requirements and design parameters. From a key characteristic point of view, this creates a map from product requirements (Product-KCs) to subsystem parameters (subsystem-KCs).

The DSM technique provides a mapping of the linkages at a high level of abstraction. The constraint-based model provides a mapping of the linkages at a detailed level. Both enable the direct mapping from product requirements to system and subsystem-KCs.

Requirement 3: Provide documentation that makes the architecture explicit and enables others to have access to the architectural knowledge (decisions).

The DSM provides one level of documentation in the form of mapping the dependencies that determine the form of the architecture. The DSM is at a high level of abstraction in order to keep the document to a "usable" length. The DSM creates the ability to communicate architectural linkages upward to management without providing all the details. It also provides a reference for design teams to know when and with whom they need to discuss proposed changes.

The constraint-based model determines the system level details for architecture. The model itself is another form of documentation that captures the detailed relationships between variables. One of the primary outputs from the model is the architecture template that can be imported into the CAD system to guide the detailed design. This template provides another form of documentation that captures the resultant layout of the system.

Requirement 4: Increase confidence in the proposed system so that product design can proceed with a minimum of risk.

Confidence about the quality of the system was improved in several ways.

From a technical point of view, the constraint-based model is solved only when the constraints have been satisfied. Thus, on this level the constraint-based model approach provides the first level of confidence.

From an organizational perspective, the use of the weekly meetings in which the results from the model were reviewed and areas of contention were highlighted and discussed increased confidence in the process. Reviewing the results on a regular basis in an open forum helped to build credibility in the process and in the results, this thereby increased confidence in the output of the process. Lastly, the documentation created by the versions of the DSM and the constraint-based model over the course of the architecture development provides a valuable history for the program team. The ability to track and review changes in this manner helps to build the confidence of program assessment teams during phase gate reviews.

This thesis suggests a methodology using design structure matrix techniques in combination with constraint-based models to develop system architecture. This approach allows detailed evaluation of proposed architectures early in the product development cycle. Shortfalls in the proposed architectures can be revised via model-based iteration while identifying key areas where system knowledge is lacking. Management can direct the team to focus their attention on filling the knowledge gaps early in the development process and funneling the new information into the architecture model. The result is a product architecture that meets the known requirements, has improved stability, and contains the documented decisions that will be required for future analysis and review.

Earlier in the thesis, Sterman's model of learning was discussed [28]. It proposes a learning model using "virtual worlds" as a means to develop and test decisions before implementing them in the real world. This enables rapid system

learning, reduces the lag time between when a decision is made and when feedback is provided on the quality of the decision, and improves our understanding of the system. With refined understanding of the system behavior the system's complex interactions become easier to manage. The information presented in this thesis suggests that the proposed methodology creates a model for learning. The framework is shown in Figure 1 has been updated to map the elements of the methodology into this framework. The updated diagram is shown in Figure 24.

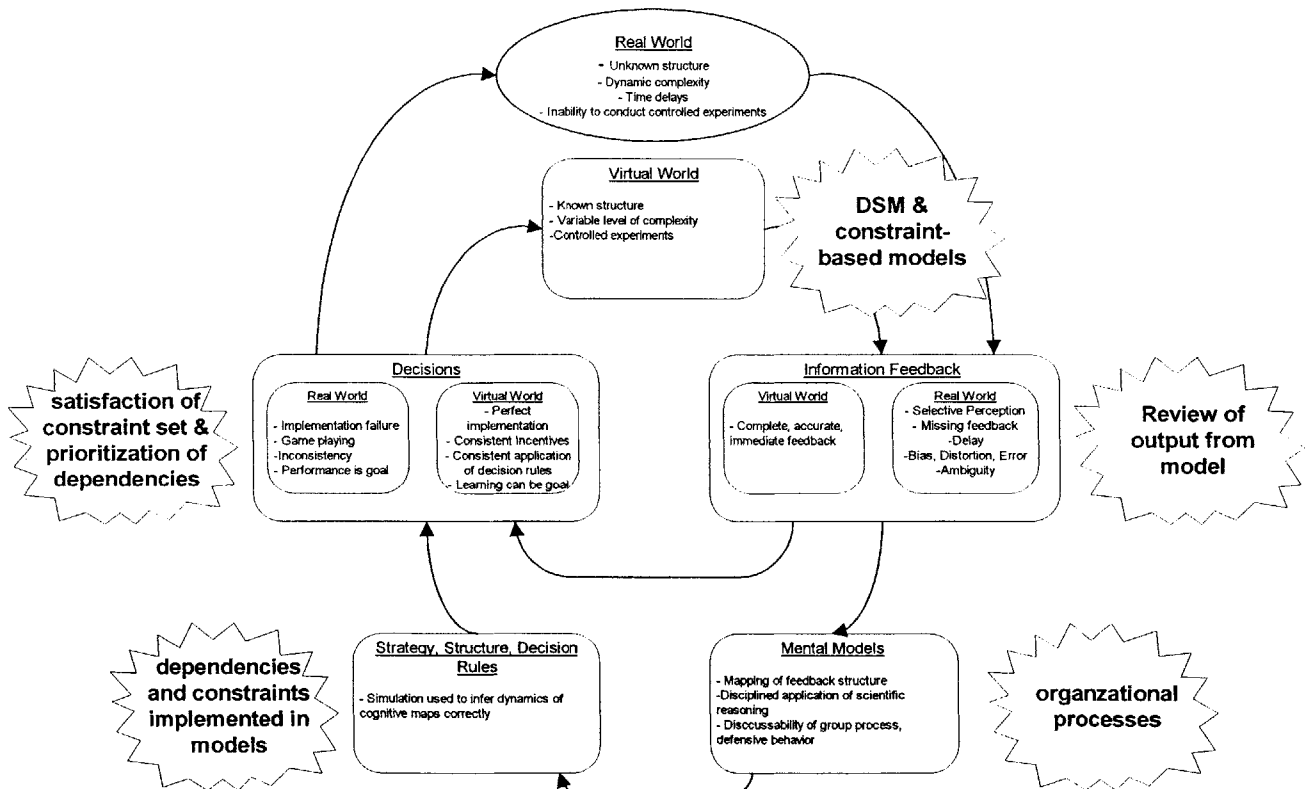


Figure 24: Methodology mapped to the model of learning

Based on the case study presented here, there is anecdotal evidence to support the hypothesis that the methodology described in this paper satisfies the needs of an architecture methodology that enables efficient system development.

---- End ----

6 APPENDIX A: PRINTING MACHINE ARCHITECTURE



**United States Patent** [19]  
**Omelchenko et al.**

[11] **Patent Number:** 5,946,533  
 [45] **Date of Patent:** Aug. 31, 1999

[54] **PRINTING MACHINE ARCHITECTURE** 5,270,769 12/1993 Satoh et al. .... 355/272  
 5,313,259 5/1994 Smith ..... 355/326

[75] **Inventors:** Mark A. Omelchenko, Lexington, Ky.;  
 Daniel W. Costanza, Rochester, N.Y.;  
 James M. Casella, Webster, N.Y.;  
 Robert M. Lofthus, Honeoye Falls,  
 N.Y.; Orlando J. Lacayo, Rochester,  
 N.Y.; Michael F. Leo, Penfield, N.Y.;  
 Michael J. Martin, Hamlin, N.Y.;  
 Joseph M. Wing, Ontario, N.Y.; Ssujan  
 Hou, Cheshire, Conn.; Michael R.  
 Furst, Rochester; Mark A. Adiletta,  
 Fairport, both of N.Y.

*Primary Examiner*—William Royer  
*Assistant Examiner*—Greg Moldafsky  
*Attorney, Agent, or Firm*—H. Fleischer; J. E. Beck; B. P.  
 Smith

[73] **Assignee:** Xerox Corporation, Stamford, Conn.

[21] **Appl. No.:** 09/212,591

[22] **Filed:** Dec. 16, 1998

[51] **Int. Cl.<sup>6</sup>** ..... G03G 15/01

[52] **U.S. Cl.** ..... 399/223

[58] **Field of Search** ..... 399/223, 162,  
 399/302, 308, 309, 364

[56] **References Cited**

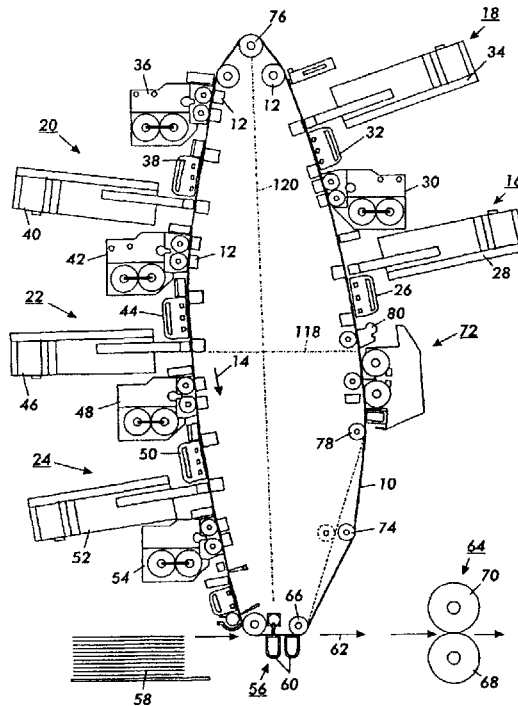
U.S. PATENT DOCUMENTS

4,998,145 3/1991 Haneda et al. .... 355/327

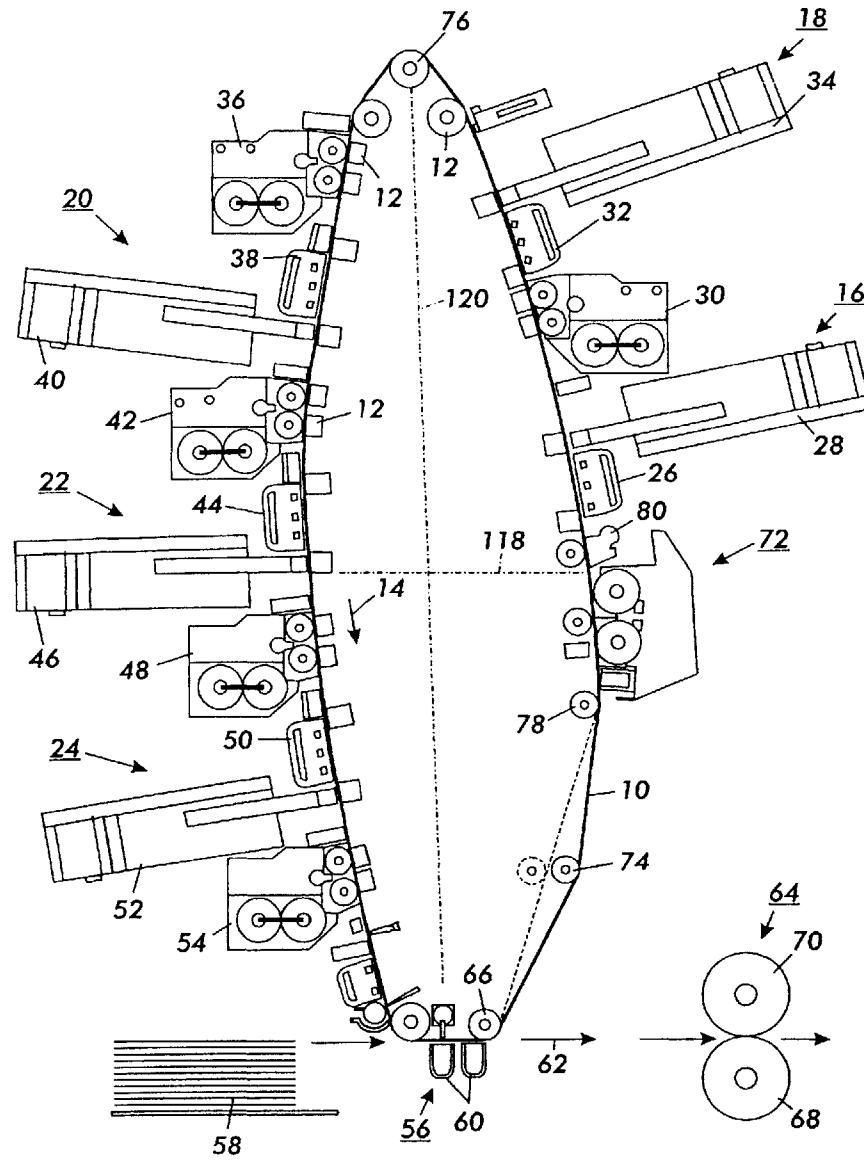
[57] **ABSTRACT**

A single pass, multi-color electrophotographic printing machine architecture uses a vertically oriented photoconductive belt. Transfer of the toner powder images occur at the lowermost portion of the photoconductive belt. The photoconductive belt is elliptically shaped, having a major and a minor axis. N image recording stations are positioned adjacent an exterior surface of the photoconductive belt on one side of the major axis thereof. N-1 image recording stations are positioned adjacent the exterior surface of the photoconductive belt on the other side of the major axis thereof. The image recording stations record electrostatic latent images on the photoconductive belt. This architecture optimizes image registration while minimizing the overall height of the printing machine.

11 Claims, 1 Drawing Sheet







## PRINTING MACHINE ARCHITECTURE

This invention relates to a printing machine architecture, and more particularly, concerns an elliptically shaped photoconductive belt having N image recording stations positioned adjacent an exterior surface of the photoconductive belt on one side of the major axis, and N-1 image recording stations positioned adjacent the exterior surface of the photoconductive belt on the other side of the major axis to record electrostatic latent images on the photoconductive belt.

A typical electrophotographic printing machine employs a photoconductive member that is charged to a substantially uniform potential so as to sensitize the surface thereof. The charged portion of the photoconductive member is exposed to a light image of an original document being reproduced. Exposure of the charged photoconductive member selectively dissipates the charge thereon in the irradiated areas to record an electrostatic latent image on the photoconductive member corresponding to the informational areas contained within the original document. After the electrostatic latent image is recorded on the photoconductive member, the latent image is developed by bringing a developer material into contact therewith. Generally, the electrostatic latent image is developed with dry developer material comprising carrier granules having toner particles adhering triboelectrically thereto. However, a liquid developer material may be used as well. The toner particles are attracted to the latent image, forming a visible powder image on the photoconductive surface. After the electrostatic latent image is developed with the toner particles, the toner powder image is transferred to a sheet. Thereafter, the toner image is heated to permanently fuse it to the sheet.

It is highly desirable to use an electrophotographic printing machine of this type to produce color prints. In order to produce a color print, the printing machine includes a plurality of stations. Each station has a charging device for charging the photoconductive surface, an exposing device for selectively illuminating the charged portions of the photoconductive surface to record an electrostatic latent image thereon, and a developer unit for developing the electrostatic latent image with toner particles. Each developer unit deposits different color toner particles on the respective electrostatic latent image. The images are developed, at least partially in superimposed registration with one another, to form a multi-color toner powder image. The resultant multi-color powder image is subsequently transferred to a sheet. The transferred multi-color image is then permanently fused to the sheet forming the color print. Hereinbefore, a color printing machine used four developer units. These developer units were all disposed on one side of the photoconductive belt with the other side thereof being devoid of developer units. A color printing machine of this type required an overly long photoconductive belt. A photoconductive belt of this type would require eleven, nine-inch pitches to operate at 100 ppm. A belt of this length will have very low yields when being made in large quantities. In addition, this results in an overly tall printing machine when the photoconductive belt is arranged with the major axis aligned vertically. The requirement of having all of the developer units or exposure stations on one side of the photoconductive belt is necessary in order to maintain image-on-image registration. Thus, it is highly desirable to reduce the overall height of the printing machine while still maintaining the required image-on-image registration.

Various types of multi-color printing machines have heretofore been employed. The following disclosures appear to be relevant:

U.S. Pat. No. 4,998,145

Patentee: Haneda, et al.

Issued: Mar. 5, 1991

U.S. Pat. No. 5,270,769

Patentee: Satoh, et al

Issued: Dec. 14, 1993

U.S. Pat. No. 5,313,259

Patentee: Smith

Issued: May 17, 1994

U.S. Pat. No. 4,998,145 discloses an electrophotographic printing machine having a plurality of developer units adjacent one another on one side of the diameter of a photoconductive drum.

U.S. Pat. No. 5,270,769 describes a printing machine having a plurality of developer units disposed on one side of a photoconductive belt. A cleaning unit is positioned on the other side of the photoconductive belt. Different colored developed images are transferred to an intermediate belt. The resultant composite multi-color image is then transferred from the intermediate belt to a sheet of support material and fused thereto. The photoconductive belt is arranged vertically.

U.S. Pat. No. 5,313,259 discloses a multi-color electrophotographic printing machine in which a photoconductive belt is vertically oriented. The machine includes four groups of stations for printing in cyan, magenta, yellow, and black. Each station includes a charged corona generator, a raster output scanning laser assembly, and a developer unit. These stations are positioned on one side of the photoconductive belt with the fourth station being disposed on the other side thereof. Successive different color toner particle images are formed in superimposed registration with one another on the photoconductive belt and transferred to a copy sheet simultaneously. Transfer occurs at the lowermost position of the photoconductive belt.

In accordance with one aspect of the features of the present invention, there is provided an electrophotographic printing machine including an elliptically shaped photoconductive belt having a major axis and a minor axis. N image recording stations are positioned on one side of the major axis and N-1 image recording stations are positioned adjacent the other side of the major axis to record electrostatic latent images on the photoconductive belt.

Other aspects of the present invention will become apparent as the following description proceeds and upon reference to the drawing, which is a schematic, elevational view showing a single pass multi-color printing machine architecture.

While the present invention will hereinafter be described in connection with a preferred embodiment thereof, it will be understood that it is not intended to limit the invention to that embodiment. On the contrary, it is intended to cover all alternatives, modifications and equivalents as may be included within the spirit and scope of the invention as defined by the appended claims.

For a general understanding of the features of the present invention, reference is made to the drawing. In the drawing, like reference numerals have been used throughout to designate identical elements.

3

Referring now to the drawing, there is shown a single pass multi-color printing machine. This printing machine employs a photoconductive belt 10, supported by a plurality of rollers or bars, 12. Photoconductive belt 10 is arranged in a vertical orientation. Belt 10 advances in the direction of arrow 14 to move successive portions of the external surface of photoconductive belt 10 sequentially beneath the various processing stations disposed about the path of movement thereof. The photoconductive belt has a major axis 120 and a minor axis 118. The major and minor axes are perpendicular to one another. Photoconductive belt 10 is elliptically shaped. The major axis 120 is substantially parallel to the gravitational vector and arranged in a substantially vertical orientation. The minor axis 118 is substantially perpendicular to the gravitational vector and arranged in a substantially horizontal direction. The printing machine architecture includes five image recording stations indicated generally by the reference numerals 16, 18, 20, 22, and 24, respectively. Initially, belt 10 passes through image recording station 16. Image recording station 16 includes a charging device and an exposure device. The charging device includes including a corona generator 26 that charges the exterior surface of photoconductive belt 10 to a relatively high, substantially uniform potential. After the exterior surface of photoconductive belt 10 is charged, the charged portion thereof advances to the exposure device. The exposure device includes a raster output scanner (ROS) 28, which illuminates the charged portion of the exterior surface of photoconductive belt 10 to record a first electrostatic latent image thereon. Alternatively, a light emitting diode (LED) may be used.

This first electrostatic latent image is developed by developer unit 30. Developer unit 30 deposits toner particles of a selected color on the first electrostatic latent image. After the highlight toner image has been developed on the exterior surface of photoconductive belt 10, belt 10 continues to advance in the direction of arrow 14 to image recording station 18.

Image recording station 18 includes a recharging device and an exposure device. The charging device includes a corona generator 32 which recharges the exterior surface of photoconductive belt 10 to a relatively high, substantially uniform potential. The exposure device includes a ROS 34 which illuminates the charged portion of the exterior surface of photoconductive belt 10 selectively to record a second electrostatic latent image thereon. This second electrostatic latent image corresponds to the regions to be developed with magenta toner particles. This second electrostatic latent image is now advanced to the next successive developer unit 36.

Developer unit 36 deposits magenta toner particles on the electrostatic latent image. In this way, a magenta toner powder image is formed on the exterior surface of photoconductive belt 10. After the magenta toner powder image has been developed on the exterior surface of photoconductive belt 10, photoconductive belt 10 continues to advance in the direction of arrow 14 to image recording station 20.

Image recording station 20 includes a charging device and an exposure device. The charging device includes corona generator 38, which recharges the photoconductive surface to a relatively high, substantially uniform potential. The exposure device includes ROS 40 which illuminates the charged portion of the exterior surface of photoconductive belt 10 to selectively dissipate the charge thereon to record a third electrostatic latent image corresponding to the regions to be developed with yellow toner particles. This third electrostatic latent image is now advanced to the next successive developer unit 42.

4

Developer unit 42 deposits yellow toner particles on the exterior surface of photoconductive belt 10 to form a yellow toner powder image thereon. After the third electrostatic latent image has been developed with yellow toner, belt 10 advances in the direction of arrow 14 to the next image recording station 22.

Image recording station 22 includes a charging device and an exposure device. The charging device includes a corona generator 44, which charges the exterior surface of photoconductive belt 10 to a relatively high, substantially uniform potential. The exposure device includes ROS 46, which illuminates the charged portion of the exterior surface of photoconductive belt 10 to selectively dissipate the charge on the exterior surface of photoconductive belt 10 to record a fourth electrostatic latent image for development with cyan toner particles. After the fourth electrostatic latent image is recorded on the exterior surface of photoconductive belt 10, photoconductive belt 10 advances this electrostatic latent image to the cyan developer unit 48.

Cyan developer unit 48 deposits cyan toner particles on the fourth electrostatic latent image. These toner particles may be partially in superimposed registration with the previously formed yellow powder image. After the cyan toner powder image is formed on the exterior surface of photoconductive belt 10, photoconductive belt 10 advances to the next image recording station 24.

Image recording station 24 includes a charging device and an exposure device. The charging device includes corona generator 50 which charges the exterior surface of photoconductive belt 10 to a relatively high, substantially uniform potential. The exposure device includes ROS 52, which illuminates the charged portion of the exterior surface of photoconductive belt 10 to selectively discharge those portions of the charged exterior surface of photoconductive belt 10 which are to be developed with black toner particles. The fifth electrostatic latent image, to be developed with black toner particles, is advanced to black developer unit 54.

At black developer unit 54, black toner particles are deposited on the exterior surface of photoconductive belt 10. These black toner particles form a black toner powder image which may be partially or totally in superimposed registration with the previously formed yellow and magenta toner powder images. In this way, a multi-color toner powder image is formed on the exterior surface of photoconductive belt 10. Thereafter, photoconductive belt 10 advances the multi-color toner powder image to a transfer station, indicated generally by the reference numeral 56.

At transfer station 56, a receiving medium, i.e., paper, is advanced from stack 58 by sheet feeders and guided to transfer station 56. At transfer station 56, a corona generating device 60 sprays ions onto the back side of the paper. This attracts the developed multi-color toner image from the exterior surface of photoconductive belt 10 to the sheet of paper. Stripping assist roller 66 contacts the interior surface of photoconductive belt 10 and provides a sufficiently sharp bend thereat so that the beam strength of the advancing paper strips from photoconductive belt 10. A vacuum transport moves the sheet of paper in the direction of arrow 62 to fusing station 64.

Fusing station 64 includes a heated fuser roller 70 and a backup roller 68. The back-up roller 68 is resiliently urged into engagement with the fuser roller 70 to form a nip through which the sheet of paper passes. In the fusing operation, the toner particles coalesce with one another and bond to the sheet in image configuration, forming a multi-color image thereon. After fusing, the finished sheet is

5

discharged to a finishing station where the sheets are compiled and formed into sets which may be bound to one another. These sets are then advanced to a catch tray for subsequent removal therefrom by the printing machine operator.

One skilled in the art will appreciate that while the multi-color developed image has been disclosed as being transferred to paper, it may be transferred to an intermediate member, such as a belt or drum, and then subsequently transferred and fused to the paper. Furthermore, while toner powder images and toner particles have been disclosed herein, one skilled in the art will appreciate that a liquid developer material employing toner particles in a liquid carrier may also be used.

Invariably, after the multi-color toner powder image has been transferred to the sheet of paper, residual toner particles remain adhering to the exterior surface of photoconductive belt 10. The photoconductive belt 10 moves over isolation roller 78 which isolates the cleaning operation at cleaning station 72. At cleaning station 72, the residual toner particles are removed from photoconductive belt 10. The belt 10 then moves under spots blade 80 to also remove toner particles therefrom.

It has been determined that belt tensioning member 74, preferably a roll, which is resiliently urged into contact with the interior surface of photoconductive belt 10, has a large impact on image registration. Heretofore, tensioning of the photoconductive belt was achieved by a roll located in the position of steering roll 76. In printing machines of this type, the image recording stations were positioned on one side of the major axis, with at most there being one image recording device on the other side thereof. Thus, there would be an image recording device on one side of the major axis of the photoconductive belt, separated by the tensioning roll, followed by four image recording devices positioned on the other side of the major axis of photoconductive belt 10. It has been determined that when the height of the photoconductive belt is reduced, requiring two image recording stations to be positioned on one side of the major axis and three image recording stations to be positioned on the other side of the major axis, image-to-image registration deteriorated. This has been overcome by changing the location of the tensioning roll so as to position it between stripping roller 66 and isolation roll 78 adjacent cleaning station 72. This configuration enabled image-on-image registration to be maintained at the same levels as a printing machine of the previous type, provided that the tensioning mechanism was interposed between stripper roller 66 isolation roll 78. Tensioning roll 74 is mounted slidably on brackets. A spring resiliently urges tensioning roll 74 into contact with the interior surface of photoconductive belt 10 to maintain belt 10 at the appropriate tension.

In recapitulation, it is clear that the present invention is directed to a printing machine architecture having N image recording stations positioned adjacent an exterior surface of the photoconductive belt on one side of the major axis thereof and N-1 image recording stations positioned adjacent an exterior surface of the photoconductive belt on the other side of the major axis. These imaging stations record electrostatic latent images on the photoconductive belt.

It is, therefore, apparent that there has been provided in accordance with the present invention, a printing machine architecture which fully satisfies the aims and advantages hereinbefore set forth. While this invention has been described in conjunction with a specific embodiment thereof, it is evident that many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, it is intended to embrace all such alternatives,

6

modifications and variations that fall within the spirit and broad scope of the appended claims.

We claim:

1. An electrophotographic printing machine, including:
  - an elliptically shaped photoconductive belt having a major axis and a minor axis;
  - N image recording stations positioned adjacent an exterior surface of said photoconductive belt on one side of the major axis thereof, whereby N is greater than one; and
  - N-1 image recording stations positioned adjacent the exterior surface of said photoconductive belt on the other side of the major axis to record electrostatic latent images on said photoconductive belt.
2. A printing machine according to claim 1, further including a plurality of developer units, with one of said plurality of developer units being positioned between adjacent said image recording stations, to develop the electrostatic latent images recorded on said photoconductive belt with different color toner to form a developed image on the exterior surface of said photoconductive belt.
3. A printing machine according to claim 2, further including a transfer station, positioned adjacent said photoconductive belt, to transfer the developed image from said photoconductive belt to a receiving medium.
4. A printing machine according to claim 3, further including a cleaning station, positioned adjacent said photoconductive belt, to remove material therefrom after said transfer station transfers the developed image to the receiving medium.
5. A printing machine according to claim 4, further including a tensioning member, positioned between said transfer station and said cleaning station and contacting an interior surface of said photoconductive belt, to maintain said photoconductive belt in tension.
6. A printing machine according to claim 5, further including an isolation member contacting the interior surface of said photoconductor belt adjacent said cleaning station between said tensioning member and said cleaning station.
7. A printing machine according to claim 6, wherein each of said image recording stations includes:
  - a charging device, located adjacent said photoconductive belt, for charging the exterior surface of said photoconductive belt; and
  - an exposure device for illuminating selected areas of the charged exterior surface of said photoconductive belt so as to discharge selected portions of the charged exterior surface of said photoconductive belt to record the electrostatic latent images thereon.
8. A printing machine according to claim 7, wherein said charging device includes a charging corona generator.
9. A printing machine according to claim 8, wherein said transfer station includes:
  - a transfer corona generator positioned adjacent the exterior surface of said photoconductive belt; and
  - a stripping member, positioned in contact with the interior surface of said photoconductive belt between said transfer corona generator and said tensioning member.
10. A printing machine according to claim 9, wherein said photoconductive belt moves in a recirculating path.
11. A printing machine according to claim 10, further including a fusing station, operatively associated with the receiving member, to fix the image transferred to the receiving member.

\* \* \* \* \*

## 7 REFERENCES

1. Browning, Tyson R., "Use of Dependency Structure Matrices for Product Development Cycle Time Reduction", *Proceedings of the Fifth ISPE International Conference on Concurrent Engineering: Research and Applications*, Tokyo, Japan, July 15-15, 1998.
2. Charman, Philippe, "A constraint-based approach for the generation of floor plans.", *Proceedings of the International Conference on Tools with Artificial Intelligence 1994.*, IEEE Piscataway, NJ, USA, pp. 555-561, 1994.
3. Cognition Corp., "Cognition's Mechanical Advantage Series Products", <http://www.ci.com/products/maprod.html>
4. Dastin, Richard M. and Castelli, Vittorio, R., "Registration Improvement by Component Synchronization in Color Printers", United States Patent 5287160, February 15, 1994.
5. Fujita, K, and Akagi, S. "Agent-based distributed system architecture for basic ship design", *Concurrent Engineering – Research and Applications*, Volume 7 Number 2, pp. 83-93, 1999.
6. Finch, William W., and Ward, Allen C., "A Set-based System for Eliminating Infeasible Designs in Engineering Problems Dominated by Uncertainty", *Proceedings of DETC'97 1997 ASME Engineering Technical Conferences*, September 14-17, 1997, Sacramento, Ca, 1997.
7. Gonzalez-Zugasti, Javier P., Otto, Kevin N., and Baker, John D., "A Method for Architecting Product Platforms with an Application to Interplanetary Mission Design".
8. Hauser, J., Godes, D., "Identifying the most effective new product development metrics", Center for Innovation in Product Development, <http://web.mit.edu/afs/athena.mit.edu/org/c/cipd/projects/godes.html>
9. Jandourek, E. Article 6: "A Model for Platform Development", Hewlett Packard Journal, August 1996.
10. Krishnan, Viswanathan, Eppinger, Steven D., and Whitney, Daniel E., Towards a Cooperative Design Methodology: Analysis of Sequential Decision Strategies, Alfred P. Sloan School of Management, Massachusetts Institute of Technology, Cambridge Mass. Working Paper 3299-91, 1991.
11. Krishnan, Viswanathan, Eppinger, Steven D., and Whitney, Daniel E., Ordering Cross Functional Decision Making in Product Development, Alfred

- P. Sloan School of Management, Massachusetts Institute of Technology, Cambridge Mass. Working Paper 3299-91, Revision 1992.
12. Kusiak, Andrew, and Park, Kwangho, *Concurrent Design: Decomposition of Design Activities*, IEEE, 1990.
  13. Kusiak, Andrew, and Wang, Juite, "Dependency Analysis in Constraint Negotiation", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No 9, September 1995.
  14. Lafon, Jean-Claude, "Solid Modeling with Constraints and Parameterised Features", *Proceedings of the IEEE Symposium on Information Visualization 1998*. IEEE Piscataway, NJ, USA, 98TB1000246, pp. 102-107. 1998.
  15. Martin, Michael, Diagram modified from System Architecture course notes, Fall 1998.
  16. Martin, Michael, Xerographic and P/R Module Architecture Modeling, Xerox Internal Presentation, Modeling and Simulation Workshop, 1995.
  17. MIT DSM Homepage, "The Design Structure Matrix – DSM", <http://web.mit.edu/dsm/index.html>
  18. MIT DSM Homepage, "Reading the DSM", [http://web.mit.edu/dsm/Tutorial/DSM\\_reading.htm](http://web.mit.edu/dsm/Tutorial/DSM_reading.htm)
  19. MIT DSM Homepage, "A Clustering Algorithm", <http://web.mit.edu/dsm/Tutorial/clustering-algorithm.htm>
  20. Naveen, Sharma, "Constraint-Based System Architecting for Xerographic Modules", *INFORMS Montreal*, April 1998.
  21. Nevins, J. L., and Whitney D. E., *Concurrent Design of Products and Processes*, McGraw-Hill Publishing Company, New York, 1989.
  22. Newbern, D, and Nolte, J., "Engineering of Complex Systems: Understanding the Art Side, *Systems Engineering*, Volume 2, Number 3, pp. 181-186, 1999.
  23. Omelchenko, M., Costanza, D., Casella, J., Lofthus, R., Leo, M., Martin, M., Wing, J., Hou, S., Furst, M., and Adiletta, M., "Printing Machine Architecture", United States Patent 5946533, August 31, 1999.
  24. Pimmler, Thomas, U., and Eppinger, Steven D., *Integration Analysis of Product Decompositions*, Alfred P. Sloan School of Management, Massachusetts Institute of Technology, Cambridge Mass. Working Paper 3690-94, 1994.

25. Portanova, P. L., and Tomei, E. J., Quality Function Deployment In Launch Operations, Contract No. F04701-88-C-0089, November 1990.
26. Rational Software Corporation, "Analysis and Design with UML", 1997.
27. Smith, Robert P., Eppinger, Steven D., Identifying Controlling Features of Engineering Design Iteration, Alfred P. Sloan School of Management, Massachusetts Institute of Technology, Cambridge Mass. Working Paper 3348-91, Revision 1992.
28. Serman, John D., Learning in and about complex systems, System Dynamics Review Vol 10 nos 2-3 (summer-Fall 1994), pp. 291-330, 1994.
29. Steward, D. V., "The Design Structure System: A Method for Managing the Design of Complex Systems", *IEEE Transactions on Engineering Management*. August 1981, pp. 71-74.
30. Steward, D., Systems Analysis and Management: Structure, Strategy, and Design, Donald V. Steward, 1995.
31. Thornton, A. C., "A Mathematical Framework for the Key Characteristic Process", accepted into Research in Engineering and Design., 1999.
32. Thornton, A. C., "Using Key Characteristics to Balance Cost and Quality During Product Development" *Proceedings of DETC'97, 1997 ASME Design Engineering Technical Conferences*, September 14-17, 1997, Sacramento Ca.
33. Thornton, A. C., "A Support Tool for Constraint Processes in Embodiment Design", *ASME Design Theory and Methodology Conference*, Minneapolis MN, September 1994.
34. Thornton, A. C. and Johnson, A. L., "CADET: A Software Support Tool for Constraint Processes in Embodiment Design", *Research in Engineering Design* 8: 1-13, 1996.
35. Ulrich, Karl T., and Eppinger Steven D., Product Design and Development, McGraw-Hill Publishing Company, New York, 1995.
36. Wood, K. L, and Otto, K. N., Chapter 9: Product Architecture, Product Design, 1999.
37. Wood, K. L, and Otto, K. N., Chapter 8: Product Portfolios and Portfolio Architecture, Product Design, 1999.
38. Yohannan, B., "Getting It Right From the Start – eProduct Architectures", *Electronic Engineering*, Volume 71, Number 872, p 3., 1999.

39. Yu, Janet S. and Otto, Kevin N., Product Architecture Definition Based Upon Customer Demands.

2976-91