1996

Computer aided development of agile tools

Ravi Jaya Shankar

Lehigh University

Follow this and additional works at: http://preserve.lehigh.edu/etd

Recommended Citation
Shankar, Ravi
Jaya
Computer Aided Development of Agile Tools

January 12, 1997
Computer Aided Development of Agile Tools

by

Ravi Jaya Shankar

A Thesis
Presented to the Graduate and Research Committee
of Lehigh University
in Candidacy for the Degree of
Master of Science

in
Industrial and Manufacturing Systems Engineering

Lehigh University
12/05/1996
This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

Date

Dec 5/96

Thesis Advisor

Chairperson of Department
Acknowledgement

I am very grateful to my parents Usha and Jaya Shankar for their blessings and guidance.

I am thankful to my advisor Prof Mikell Groover for his guidance and providing an opportunity to work on this project. I am thankful to my team members Prof Manash Ray, Prof Tulga Oszoy, Albert Poe, Steve Roth, Mary Meixell, Erkan Baykan and Doo Hwan Kim for their timely help and suggestions.

Without my friends Debbie, Balu, Vicky, Michael and others, I wouldn’t have been typing this sentence. Thank you all for the great support.

This material is based on work supported by the U.S. Air Force under Contract No F33615-94-C-4428 and/or F33615-94-C-4429. Any opinions, findings, and conclusions or recommendations expressed in this report are those of the author and do not necessarily reflect the views of the Air Force. It is published for the exchange and stimulation of ideas.

Finally I thank Almighty God for showering His/Her grace on me.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgement</td>
<td>iii</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>iv</td>
</tr>
<tr>
<td>List of Illustrations</td>
<td>vi</td>
</tr>
<tr>
<td>Abstract</td>
<td>1</td>
</tr>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Transactions Analysis</td>
<td>3</td>
</tr>
<tr>
<td>Objective</td>
<td>4</td>
</tr>
<tr>
<td>Agile Tools</td>
<td>5</td>
</tr>
<tr>
<td>Theory of Transactions Analysis</td>
<td>5</td>
</tr>
<tr>
<td>Transaction Maps</td>
<td>7</td>
</tr>
<tr>
<td>Design Structure Matrix</td>
<td>9</td>
</tr>
<tr>
<td>Graph Structure</td>
<td>12</td>
</tr>
<tr>
<td>Software Development</td>
<td>14</td>
</tr>
<tr>
<td>Introduction</td>
<td>14</td>
</tr>
<tr>
<td>Motif</td>
<td>14</td>
</tr>
<tr>
<td>Object Oriented Programming</td>
<td>15</td>
</tr>
</tbody>
</table>
List of Illustrations

1) Snapshot of a data flow diagram ........................................ 6
2) Typical transactions map.................................................... 8
3) Information flow relationships between tasks...................... 10
4) Design Structure Matrix (DSM).......................................... 11
5) Flow chart for the DSM creation process......................... 13
6) Creation of TA maps using the software.............................. 18
7) Software representation of the DSM................................. 20
Abstract

A fast and flexible product development process is crucial to reduce the product cycle time and overhead costs. Tools are developed to plan, design and manage the development process. There are numerous activities extending over many organizations as part of the process. Information flow between these activities determines the speed of development. Tools based on "Transactions Analysis" are developed to capture, represent and analyze the actual information relationships. Software with a graphic user interface is developed with the objective of automating the application of these tools. The tools are applied to real industry data and the scalability and migratability of the tools is proved. The software developed is fast and inter-operable.
Introduction

The nations of the new world order are competing for economic supremacy in an open market field with many competitors. The companies of these nations are multinational. They compete with companies from different nations, technologies and cultural composition. In this new agile environment the key to success is to deliver high quality products in a short quantity of time at minimum cost [2]. The cycle of conceiving the product and delivering to the customer is called the concept to cash cycle. Two main aspects of this cycle are product development process (PDP) and product manufacturing process. Conceptualizing the product, determining the demand, designing the product specifications, checking manufacturing feasibility and other important activities are performed by the company before starting the manufacturing process [5]. Such activities are part of the product development process. The time taken by companies to realize and develop a product is a large part of the product cycle time. The process involves a complex set of activities and a large overhead cost for the company. Reduction of the product development process time is an important objective for the company.

Competitors introduce new products to meet the fast changing demand. The company has to be fast and flexible as a consequence of this intense competition. Flexibility for a company is determined by the adaptability of its product development process for new products [1]. A direct consequence is that the reduction of PDP time is the top priority.
A crucial feature is that, in every industry, product development is not an activity of one company but is in reality a joint activity of many firms that form a "supply web." Many of the problems encountered during PDP are because shared development is not properly managed or planned [4].

Supply web management consists of several activities:

- Attention to technical coordination with specific agreed-upon definitions of engineering data, specifications, and reference points such as geometric data.
- Establishment of organizational arrangements that manage the distributed responsibility for delivering the technical specifications.
- Definition of incentives and relationships that permit the web to operate successfully during problem situations.

The mapping of various activities of the actual product development process across the supply web is critical to manage or plan the PDP. An effective and important tool adapted to achieve this purpose is called Transactions Analysis [6].

**Transactions Analysis**

Transactions analysis (TA) are interview-based studies of how organizations operate. Performing transactions analysis in various industry sites [6] shows the inherent complexities of engineering partnerships and is used to develop the information flow architecture of the product development process. Transactions analysis reveals intensive transactions activity areas and also permit one to see how activities at one point in the
process are linked to activities elsewhere. Actual transactions do not correspond to official organization charts or approved information transfers, and the degree to which they differ is a good indication of how the participants, of this analysis, must change the official process to remain competitive.

**Objective**

This research is part of the MIT/Lehigh PATHFINDER project. The primary objective of the project is to develop agile tools for planning and managing the PDP [8]. The project identified transactions analysis as an effective tool for data collection and representation. The transactions maps developed from the results of transactions analysis were represented by data flow diagrams only as a hard copy. This prevents a person using these maps to make changes or perform analysis. A graphical software interface enables a flexible method of representing the data and performing analysis. The transactions maps are converted to other tools such as Design Structure Matrix. Automation of the conversion process is desirable to handle a large number of activities, which are typical of real industry data. The research aims to develop software tools for representation of the transactions analysis data and provides utilities for performing analysis on the representations. The tools developed are implemented at the site of FORD, North Penn Electronics Facility (NPEF), Lansdale PA. The other objective is to prove that the tools developed are scalable and migratable.
Agile Tools

Theory of transactional analysis

In addition to being a theory of development, Transactional Analysis is a system of social psychology developed by Eric Berne, M.D. in the 1960s while he was studying to be a psychoanalyst [9]. As a social psychology, Transactional Analysis looks at the relationships within the ego and between individuals. TA has quickly become a popular resource in psychotherapy, education and organizational development because it helps people to see the many sides of their own personality and understand how their personality relates to others. As a culture, people are absorbed in individual psychology and often fail to understand the interconnected nature of human beings. TA is a theory and practice of psychology that recognizes the basic interdependence, interrelatedness and difficulties which arise with dependence and cooperation. In this sense, Transactional Analysis is the psychology of the 21st century.

In this research transactional analysis is termed "transactions analysis" and uses the essential techniques from the social method for personality development. The transactions are considered as information flows between activities. These activities are part of the product development process and the supply web. The results of transactions analysis are documented by tools such as Transactions Maps and Design Structure Matrix.
Figure 1. Snapshot of a data flow diagram from a real PDP.
Transactions Maps

These maps are basically data-flow diagrams. One version of a typical data flow diagram is shown in Figure 1. The data from transactions analysis is analyzed and converted to data flow diagrams. The diagram in Figure 2 shows a typical transactions map. The nodes represent the activities and the lines with arrowheads, called arcs, represent information flow. The entire PDP of a company can be represented in such transactions maps. The activities are part of a process tree or occur simultaneously with many activities. Parallel activities are represented by parallel maps. Figure 2 shows serial and parallel activities. A basic rule followed in drawing transactions maps is that horizontally the activities occur sequentially and vertically activities occur simultaneously. Figure 2 shows feedback flows which break this rule and introduce complexity in the map representation.

Transactions maps are powerful to analyze sequential and parallel activities. However informational relationships among activities are poorly represented in this method. The complexity introduced by feedback flows cannot be analyzed from this representation. A tool which captures complexities and relationships in the information flow is necessary. Design Structure Matrix provides a graphical and analytical representation to achieve this objective.
Figure 2. Typical transactions map
Design Structure Matrix

The information flow representation must include not only the sequence of activities but also the many technical and informational relationships (Figure 3) among the activities. The words "task" and "activity" mean exactly the same in this research. This is achieved using a description called Design Structure Matrix (DSM) [3]. The DSM is especially suited for identifying clusters of related activities as well as pointing out places where the process is likely to repeat a chain of activities. Figure 4 shows a DSM in its simple form represented by identically labelled row and columns. The X marked elements in each row identify which other activities must contribute information for the completion of the design process. For example, the X marks in row 4 are in columns, showing that activity 4 required information from activities. These activities are desired to be performed before activity 4. Thus, the first step is to find a sequence of these design process activities which allow the matrix to be lower triangular. If the activities can be sequenced so that each one can be executed only after it receives all the information it requires from its predecessors, then no coupling remains in the process. This rarely happens. Typical DSMs capture all the coupled activities. These activities are grouped into blocks containing all coupled activities and identified as departments or groups. The X marks in the upper triangle, which are not blocked are the feedback flows. These flows increase the product design process time and reduce the flexibility of the PDP. The DSM is used to identify feedback flows and redesign the PDP to reduce their number.
Figure 3 Information flow relationships between tasks.
Figure 4. Design Structure Matrix for the figure 2.
The transactions maps described in the previous section represent the information flow in the PDP. These maps are converted into DSMs to capture the information relationships, feedback flows and activity clusters existing in the PDP. The maps involve sequential activities and parallel sequences. Initial judgement is required to resolve the sequence of parallel activities because the DSM represents activities in the time sequence.

The general procedure for transactions analysis, generation of transactions maps and developing the DSM is described in the flow chart in Figure 5.

**Graph Structure**

The transaction maps are essentially a graph structure. The activities form the nodes and the information flows form the arcs of the graph. The graph is directed as the flow is always in one direction. The basic structure is a tree with time sequence determining the hierarchy on nodes. Feedback flows form closed paths and hence convert the tree to a directed graph. The DSM is 2-dimensional matrix representation of the same graph structure preserving the same hierarchy. The rows and columns represent the nodes of the graph and the marked elements the arcs. The increasing indices of the rows and columns determine the time sequence. Conversion from one representation to the other is possible as the underlying data structure remains the same.
Figure 5. Flow chart for the DSM creation process.
Software development

Introduction

The main objective of this research is to develop software tools which require minimum user interaction. The primary users of this tool are managers, designers and planners. A graphic user interface (GUI) is a necessity for achieving this goal. The product development process is represented graphically by transactions maps and DSM. This interface should provide utilities to create, edit, save and perform analysis. The GUI chosen is windows based and provides move and click mouse interaction. This project was developed on AIX operating system using IBM RS6000 machines. The GUI was developed using Motif and X-Windows libraries. X-Windows is a window manager based on Unix operating system. This provides X, Xt, Xlib libraries for event, window management and GUI development.

Motif

Motif is a library of routines that make the creation of user interfaces in an X Windows environment fairly easy and straightforward. The Motif libraries handle most of the low-level event handling tasks common to any GUI system, so that sophisticated interfaces can be created without having to contend with the complexity of the operating system X Windows.

Motif is a widely-accepted set of user interface guidelines developed by the Open
Software Foundation (OSF) around 1989 which specifies how an X Window System application should "look and feel". OSF/Motif, as it's more formally called, includes the Motif Toolkit (also called "Xm" or the "Motif widgets"), which enforce a policy on top of the X Toolkit Intrinsics ("Xt"). Xt is really a "mechanism not policy" layer, and Xm provides the specific "look and feel"[10].

Motif itself is described as a "widget set". A widget is simply a user interface object - a pushbutton, a scroll bar, a text area, a list box, etc. Applications are built from collections of widgets. The widgets collect input from the user. Motif is a collection of 40 or so widgets and is an "event driven programming environment". This means that the user interface is programmed to respond to user "events". When the user hits a key on the keyboard, that generates an "event" that the program must then respond to. When the user moves or clicks the mouse, events are also generated. An "event loop" sits and waits for user events. When events occur, the event loop calls the correct piece of code to handle them.

Object Oriented Programming

The basic concept in this approach is that of objects, which consist of data structures encapsulated with a set of routines, often called "methods" which operate on the data. Operations on the data must be performed via these methods, which are common to all instances of objects of a particular class. Thus, the interface to objects is well defined,
and allows the code implementing the methods to be changed so long as the interface remains the same. This briefly defines the essence of object oriented programming.

The same concepts are used in Motif and the software developed as part of this research. The objects are the widgets like scroll bars, menu bars, nodes, arcs, events, matrix elements and utilities. The methodology of data encapsulation and inheritance is used in Motif to build extensive libraries based on X, Xt and Xlib libraries. The software incorporates the same technique and classifies objects like nodes, arcs and matrix which are not dependent on the interface of the user. New methods and utilities can be added or existing implementations changed without changing the interface. Thus, the primary benefit of this technique is the flexibility of the software without affecting the core part of the programs. Customizing the software to a particular user reduces to adding a few lines of program without changing the existing software.

There are two modules in the GUI each with a separate window:

- Transactions Map
- Design Structure Matrix

The Transactions Map Window

The transaction maps described in the previous sections are represented in this window. The maps are shown as a graph. Each activity is a node and each transaction is an arc. The arcs are directed arrows showing the direction of information flow. There are two
ways of creating the map:

- Accessing the database to open existing maps
- Interactively generating the graph in the window

A sample transaction map is shown in Figure 6. The nodes are seen as rectangular blocks and arcs are directed arrows. A double click on any node brings up a dialog box which contains the name of the activity. The name is edited or the default is saved.

The utilities in the transactions map window are:

- The edit utility contains editing operations such as add, delete, move and change properties for nodes and arcs.
- The save utility converts the graph in the window to a text file. This file forms the database for the transactions map represented by this graph.
- The analysis utility converts the graph into a design structure matrix. A new window is brought up to show this module.

There are two applications for this module:

- The actual product development process- The TA data are converted into transaction maps using the module provided to interactively generate graphs.
- Design of new processes- New transaction maps for changing the PDP are developed using the technique of interactive graph generation.
Figure 6 Creation of TA maps using the software
The Design Structure Matrix Window

The DSM is created only from an existing transactions map. A sample DSM created is shown in Figure 7. The names of the rows and columns are obtained by double clicking on their labels. The principal diagonal of the matrix is represented by black cells. The arcs are represented by color cells.

The utilities in the DSM window are:

• The edit utility allows the user to change the sequence of activities. The software automatically changes the DSM. New information flows or arcs can be added in the cells.

• The save utility converts the DSM to a text file. This file forms the database for the graph representing the DSM.

• The analysis utility finds the number of activity clusters, feedback flows or converts the new DSM into a transactions map.

Database Representation

A graph is a set of nodes and arcs. The arcs connect the nodes. The arcs are directed in one direction. This is the underlying structure of the transaction maps and DSM. The transaction maps are essentially flow diagrams. The DSM is the from-to structured matrix representation of the flow diagrams. The graphs are stored in the databases as text files.
Figure 7 Software representation of the DSM
Figure 7 Software representation of the DSM
There are two types of files.

- Node file
- Arc file

Each line in the node file contains the following items

- Node index
- Node name
- Node position

Each line in the arc file contains the following items

- Arc index
- From - node index
- To - node index

The database is accessed by both the modules to show the maps and DSM. Editing operations on the database can be performed from both windows.

The Algorithm

The primary objective of the software tool is to create a graphical representation of transactions map, DSM and provide interconvertibility. In the previous section it is shown that both the representations have the same underlying data structure.

The method of interaction and the utilities provided with each tool is described in the earlier section. In this section the algorithms involved are described.
Create Graph from file or window

- Create Nodes and Arcs
- Edit Nodes and/or Arcs
- Edit Graph
- Save Graph to file

Create DSM from saved Graph file

- Node names in first column (show to-activity)
- Node names in top rows (show from-activity)
- Create cell colors
- Edit cells
- Perform Analysis
- Save DSM to file

The software does not depend on any complex algorithms which are time dependent. Simple subroutines for sorting, counting, X Window routines, file management are used. Algorithmic efficiency is not of concern because the most complex routine is of $n \log n$ where $n$ is the number of activities. The interface used is event driven, hence the time taken for the software to complete different utilities depends on the operating system.

**Agility of Software Tool**

The tools used to implement the concepts of agility in any product development process should have an agile nature themselves. The specifications for such a tool are:

- Scalable
• Migratable
• Fast
• Flexible

These specifications are achieved for both TA maps and DSM as proved in the section on "conclusions". The next step is to make this software achieve these requirements. All the above specifications are achieved by this software because of the following reasons:

• Scalability is tested based on the ability to represent and handle different sizes of data. The software handles data from a minimum of 1 activity to a maximum of 500 activities. Representation of PDPs which have higher than 500 activities are split into many TA maps and the DSM windows are linked to show the entire process.

• Migratability is tested based on the ability to apply this tool for different organizations, supplier chains, departments and processes. The software handles different types of activities, flows and processes because of the object-oriented techniques used to support generality. Any activity and flow reduces to a graph structure with nodes and arcs. The software uses a uniform data structure for any process and hence is migratable.

• Speed of the software is tested based on the time taken to perform a specific application. The software includes only polynomial time algorithms and the number of operations are dependent on the number of activities. The time taken to perform any utility depends on the operating system and its ability to perform
because of the event driven graphical interface. For the real data obtained at the sites, user interaction time was the significant factor in the total software performance time. The CPU time used was small compared to this factor. This fast feature is very useful in the analysis of large PDPs.

-Flexibility of the software is tested based on its inter-operability across different platforms. The programming was done in C language using Motif libraries on X Windows. All these are international standards and are provided by all computer manufacturers. Thus, the performance of the software on any workstation running X Windows is the same. X servers are emulated on Windows 95 and Windows NT using XWin-32 emulators. The software was successfully tested on a PC using the Micro X emulator. These reasons prove inter-operability of the software tool and the graphic interface across different platforms.

**Site Implementation**

**Introduction**

NPEF is a FORD plant and manufactures electronics parts for FORD products. NPEF is an automotive electronics build-to-print manufacturer working in a state-of-the-art facility. Among the products they manufacture are engine controllers and sensors. Ninety percent of NPEF's products are directly or indirectly supplied to FORD. They are
beginning an effort to garner a larger percentage of non-FORD business. Fundamental to this strategy will be creating an engineering design capability in-house. Transactions analysis and other agile tools for mapping and diagnosing the product development process are applied at this site.

Implementation

The transactions analysis was performed by Mary Meixell, Albert Poe and Steve Roth of Lehigh University. Steve and Albert converted the NPEF transactions analysis data into transactions maps. These maps were created as hard copies of data flow diagrams. The first part of this research was to convert the transactions maps into DSMs. Three sets of maps were considered:

- The PDP II manual of NPEF [7].
- The TA maps for the activities of the higher level phases.
- The TA maps of the activities for NPEF.

The DSM for each of these sets of maps are created using the techniques described in the previous section. The PDP II manual is elaborate and contains a large number of activities. NPEF follows this manual for the engine controller development process. The activities are grouped into a higher level of abstraction and the total number of activities are reduced from 500 to 140. The other map sets do not require this conversion as the level of abstraction of the activities in the TA is appropriate for analysis.
Results

The DSM is represented as a spreadsheet using MS EXCEL. The spreadsheet, being a graphical representation of a matrix requires data entry of the activity names and interactive placement of the X marks depicting information flow between activities. The output is obtained on a plotter which draws out the DSM on a single sheet. This is a time-consuming activity and involves manual grouping of activity clusters. The only method of analysis is visual inspection by an expert. However the spreadsheet is a very good editing tool.

The software developed as part of this research is used to achieve the following objectives:

- Automatic generation of NPEF TA maps from existing databases with minimum interaction.
- Representation of NPEF TA maps with a Graphic User Interface(GUI) with editing utilities.
- Automatic conversion of TA maps to DSM with interaction to resolve parallel activity maps.
- Representation of DSM with GUI with editing utilities.
- Analysis tool for DSM.

Some parts of the spreadsheet results are shown in Appendix A. The software windows are shown in Appendix B. The complete DSMs of NPEF are not included as part of this
thesis to maintain data confidentiality.

The following DSMs are generated from the transactions maps:

- The DSM of the PDP II manual of NPEF showed few feedback flows. Most of the feedback flows were part of the same department or activity block.
- The DSM of the TA are separated into two parts:
  * The part of the development process which occurs outside NPEF. The transactions are grouped into phases and this is seen in the DSM. The feedback flows exist inside the individual phases.
  * The part of the development process occurring inside NPEF show very few feedback flows which are not part of any groups.

These observations show that the PDP manual is faithfully followed by NPEF and the groups involved with only very few extraneous information relationships.

Conclusions

The product development process involves a large number of coupled activities. Iteration and feedback are an inherent characteristic of design activity. The DSM is a tool to analyze the information relationships between activities. The DSM is used for
resequencing the activities to reduce the design process time. It is also used as a display of the entire design process or of the designer's view of the process. The couplings and unnecessary delays are graphically seen in the matrix.

The actual design process data for NPEF, Lansdale is obtained from transactions analysis. This is represented as a transactions map using the software. Transactions maps are good tools for representation of parallel activities. The DSM represents activities in time sequence. The generation of DSM from TA maps requires judgement to resolve parallel to sequential conversion. The DSM was generated using a spreadsheet and by automatic conversion of the TA maps using the software.

Few feedback flows in the DSMs show that the manual is followed faithfully. The primary advantage of this approach is to generate very good exit reports at the completion of each phase in the PDP. This requires that a large number of checks are performed within groups of activities to maintain good phase exit reports. A direct consequence of this practice is the increase in time taken for each group of activities, increasing the total product cycle time. The manual is produced by agencies outside NPEF and designing a new PDP requires complex activities and a longer time. This makes it difficult for NPEF to compete for new non-FORD business. The PDP has to be generated in-house to acquire new business and decrease product cycle time. The methods of TA and DSM are shown to be effective tools to analyze, plan and manage the product development process.
In the earlier part of the Pathfinder Project the DSM was generated for the GM Saginaw plant [11] with great success. The size and scope of operations of the GM plant are large compared to NPEF. A comparison of the two plants is shown in Appendix C. This proves that DSM is migratable across different organizations and companies. The DSM is applied to various levels of activity clusters, departments and the entire supply web. This proves that DSM is scalable across different sizes and structures of organizations. The research objective to show that transactions analysis, transactions maps and DSM are scalable and migratable is satisfied.

The software tool developed is easy to use and provides a graphical user interface. Thus, the learning time for an average person using this software is minimized. The tool can be used for research, design and analysis by experts, managers, designers and planners. The software is robust and flexible due to the object oriented design. Interoperability is a salient feature of the tool as it is developed using universal libraries.

Future research

The areas for future research are:

• Presently DSM represents only information flows in the PDP. The next step is to include costs and probabilities in each cell of the DSM. The relative strength of each transaction can be included to capture the importance of certain transactions.
• Identifying key activities in the PDP. When the DSM is optimized to reduce the number of feedback flows the key activities should not change their hierarchy. The judgement stage involved in resolving parallel flows can be removed by identifying the key activities. Thus, this tool can be automated completely.

• Incorporating more analysis utilities so that the tool can be used as the main interface to design and analyze the PDP.
Bibliography


Appendix A
<table>
<thead>
<tr>
<th>DSM of PDP Phases for NPEF</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive HCR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identity new components</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial release to WERS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial PWB layout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop initial PWB layout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop initial product spec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identity new components</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capture schematic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Issue plant fac &amp; tooling P05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approve mg equipment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design mg process requirement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design mg equipment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define mg process requirement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test PH2 prototype</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build PH2 prototype</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update HCR PH2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conduct PH2 PEAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop PEB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop PEB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update WERS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review PWB layout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write WDR (low chain)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update BOM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product spec update</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perform FMEA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update HCR PH1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conduct PH1 PEAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test PH1 prototype</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build PH1 prototype</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build PEB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source new components</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial release to WERS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial PWB layout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop initial product spec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identity new components</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capture schematic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Issue plant fac &amp; tooling P05</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend: X = Task completed
         - = Task not applicable
Menu bar for the Transaction Map

Input shell for transaction data
INTENTIONAL SECOND EXPOSURE

Menu bar for the Transaction Map

Input shell for transaction data
Picture of the DSM window
Picture of the DSM window
## Appendix C

**North Penn Electronics Facility (NPEF)**

Ford Motor Company, Lansdale PA

<table>
<thead>
<tr>
<th></th>
<th><strong>SAGINAW (GM)</strong></th>
<th><strong>NPEF (Ford)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Plants</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Employees</td>
<td>13,800</td>
<td>2,000</td>
</tr>
<tr>
<td>Product</td>
<td>Axle and steering components</td>
<td>Electronic control devices</td>
</tr>
<tr>
<td>Processes</td>
<td>Machining, forging, assembly</td>
<td>Electronics assembly</td>
</tr>
<tr>
<td>Other</td>
<td>Design engineering located on-site</td>
<td>Design engineering located off-site</td>
</tr>
</tbody>
</table>
Ravi Jayā Shankar was born on June 24, 1973 to his parents T S Jayashankara and Usha Jayashankara in Bangalore, India. He graduated from Indian Institute of Technology in Madras, India with an undergraduate degree in Mechanical Engineering in May 1995.

He graduated from Lehigh University, Bethlehem, USA in January 1997 with a Masters degree in Industrial Engineering. He plans to make the world a better place.
END
OF
TITLE