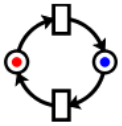

System Architectures Laboratory

Cluster Tool Simulation: Implant Source Life and Lot Scheduling

Cluster Tool Modeling Group

Claire Gryphon
Muhaimeinul Khan
Tariq Hanif
Luis Ayala

George Mason University
Systems 699 – Fall 2012

Table of Contents

1 Executive Summary

The use of semiconductors has grown through the years and can be seen everywhere from computing, networking, and server applications, to mobile, embedded, consumer, automotive, and industrial design. A key component to semiconductors development is the use of cluster tools. Cluster tools are systems used to process microelectronic devices. They are composed of processing modules and electronic arms that are controlled by a centralized system. The key advantages of these tools are that they reduce contamination and allow for the processing of wafers in bulk. The work of the Cluster Tool Modeling Group (CTSG) centers on a specific cluster tool: the implant tool, with an interest on the effect recipe sequencing (the way a group of recipes are scheduled for processing) has on the tool's ion source.

Implant tool scheduling systems try to process lots (groups of wafers) in the shortest possible amount of time. However, they often do not take the ion source's life into consideration when developing the processing schedules. Instead of learning from experience, which can be expensive, using a simulation tool could help the client test recipe schedules before applying them to the real tool. This way the availability of the implant tool can be extended, thus, resulting in less time spent on tool maintenance.

In order to approach this problem, the Cluster Tool Analysis Group applied systems engineering methods to model the ion implantation process for an implant tool with ion source deterioration. Different scheduling algorithms were tested in the model of the implant tool in order to simulate the implant process and its effect on the life of the ion source. This was done to demonstrate the usefulness of the model.

The project initially included the identification of recipe sequences to extend the life of the ion source. However, this objective was removed due to lack of data for analysis, and a tight schedule for project completion after Micron, our previous sponsor, decided not to continue with this project. There were key road blocks encountered in obtaining data as our initial project sponsor, could not provide the data because it was considered too sensitive to be provided to external users. Therefore, our new client, Systems Architectures Laboratory (SAL) at George Mason University requested that develop a simulation model that is generic enough to allow users to input their own data.

Our recommendations are intended to be implemented with minimal effort or future assistance by any entity using implant tools for processing wafers. Our identification and implementation of the appropriate systems engineering processes can also be easily replicated and re-usable. Furthermore, the model can be modified to simulate other processing tools.

2 Sponsor

Our initial sponsor was Micron Technology. However, we had to change our client because the legal procedures needed for them to provide us with data would take too long to allow us to complete the project on time. As a result, the Systems Architectures Laboratory (SAL) at George Mason University offered to be our client. SAL conducts research in the modeling, design and evaluation of architectures for information systems, and they are currently working with Micron Technologies on cluster tool modeling research, which closely relates to our project.

3 Problem Definition

3.1 Background

Cluster tools are systems used in the semiconductor processing industry to fabricate microelectronic devices and components. One of the advantages of cluster tools is that they can perform processes in sequences to improve product yield. Furthermore, they reduce contamination, which is very important in the processing of semiconductors. All semiconductor processes depend on these tools to process wafers in their respective areas.

One such cluster tool, known as an implant tool, was the object of interest for this project. Implant tools are composed of several hardware components with the main internal component being the ion source. The ion source is composed of two main parts: a cathode and a filament. The filament heats up the cathode, which in turn interacts with the gas flow; thus, creating a plasma. The plasma is then guided through magnets, acting as ion filters, and is then accelerated to implant the filtered ions onto the surface of the wafers.

In the implant process, wafers are bombarded with ions from different elements – called dopants – such as, Boron (B), Phosphorous (P), Arsenic (Ar), Carbon (C), and Germanium (Ge) into and on top of a wafer in order to modify its conductivity. The specific format used to implant these elements is known as the implantation recipe. The recipe contains information, such as the amounts of gases, pressure, temperature, and current needed for the implantation process.

The implantation recipe is critical in helping to optimize the throughput of implant tools, which is an activity of great interest today with the continuously decreasing size of semiconductors. One method in particular that is used to improve product yield is Recipe Sequencing. This method can reduce the tool preparation time between different implantation recipes by organizing the recipes in sequences, where the wafer batches are scheduled to be processed depending on the implantation recipe they require, instead of on a first come, first serve basis.

3.2 Problem statement (SOW)

In implant tools, the interactions between the gases, the heated cathode and filament, and the plasma, affect the thickness of the source's cathode. Certain gases, such as Boron (B), deposit or grow layers of substances on the surface of the cathode; other gases, such as Arsenic (Ar), Phosphorous (P), Carbon (C), and Germanium (Ge), tend to erode the surface of the cathode, making it thinner. Thus, running a process with the same element for too long may lead to early failure of the ion source.

If the cathode of the source component becomes too thin, the plasma may have direct contact with the source's filament, leading to source failure. Similarly, if there is too much build up on the surface of the source's cathode, the uniformity of the ion beam is deteriorated, leading to defects on the implanted wafer. In either situation, the source needs to be replaced, which can result in hours of lost productivity.

Current recipe sequencing doesn't take source deterioration into account, which results in frequent source changes and a potentially less than optimal throughput. Therefore, the deterioration of the ion source should be taken into consideration when organizing recipe sequences for the implantation tool. If tool availability could be extended by modifying recipe schedules, unnecessary delays could be prevented, and the throughput of the implant tool may be improved.

3.3 Scope

The Cluster Tool Simulation Group was asked to model the behavior of one implant tool to simulate the implant process for different recipe sequences with deterioration. The development of our simulation tool focused on two areas:

- The effect of different recipe sequences on time to completion for lot processing
- The effect of different recipe sequences on the life of the ion source.

3.4 Objectives

The expected outcomes of this project were the:

i. Application of a systems engineering approach for cluster tool modeling.
ii. Development of a model that can simulate an implant tool with ion source deterioration.

3.5 Success Criteria

CTSG considered the project to be successful when objectives i, and ii had been achieved and the simulation tool had been validated (the tool was proved to behave like a real implant tool).

4 Requirements

4.1 Project Requirements

1) CTSG shall use UML to model the external operation of an implant tool.
2) CTSG shall develop scheduling systems to test in the simulation tool.
3) CTSG shall model the ion source deterioration effects as random distributions.
4) CTSG shall use Colored Petri Nets (CPN) to develop an executable model to simulate the implant process with deterioration of the ion source.
5) CTSG shall demonstrate the use of the simulation tool with the developed scheduling systems.
6) CTSG shall provide recommendations to the client about the use of the tool.
7) CTSG shall produce a final report to be presented to stakeholders on December 7th 2012.
8) CTSG shall develop a website containing all documentation for this project.

4.2 Functional Requirements

1) The simulation tool shall allow the user to input recipe information through a text file.
2) The simulation tool shall allow the user to change the ion source deterioration effects.
3) The simulation tool shall keep track of the setup time of lots processed.
4) The simulation tool shall keep track of the recipe times of the lots processed.
5) The simulation tool shall keep track of the total processing time.
6) The simulation tool shall keep track of the number of source changes during a run.
7) The simulation tool shall keep track of the number of lots processed during a run.
8) The simulation tool shall keep track of the number of lots that were not processed during a run due to source failure.
9) The simulation tool shall keep track of the life of the ion source as lots are processed.
10) The simulation tool shall keep track of the sequence of lots provided by the user.

11) The simulation tool shall warn the user if the sequence provided was not completed due to ion source failure.
12) The simulation tool shall return an output report with ion source status, # of lots processed, # of lots unprocessed, # of source changes, and warnings.

5 Approach

5.1 Overview

The Cluster Tool Simulation Group (CTSG)'s approach was to develop a simulation tool to allow users to test different schedules for recipe sequencing before they are implemented in a real implant tool. This way, users can get an estimate of when the ion source will fail or if it will fail before they have completed a sequence of lots. As a result, the simulation tool may allow users to identify whether there's a need for improvement in their scheduling methods.

The simulation tool was designed to read recipe information, lot information and ion source deterioration data from a text file, and to use this information to simulate the implantation process for given recipe sequences (from a scheduling algorithm). The tool was also designed to return a simulation report containing the status of the ion source's life, the total processing time for the given recipe sequence, and a warning if the ion source failed before the sequence was completed.

CTSG also developed scheduling algorithms to validate that the model behaves like a real implant tool and to perform a sample analysis of scheduling systems for scheduling improvement. The purpose of the sample analysis was to demonstrate the usefulness of the simulation model.

The process for the simulation is depicted in figure 4.1. The scheduling system generates a recipe sequence to be tested in the simulation model. The simulation model takes an input file and the recipe sequence as inputs, and returns an output report.

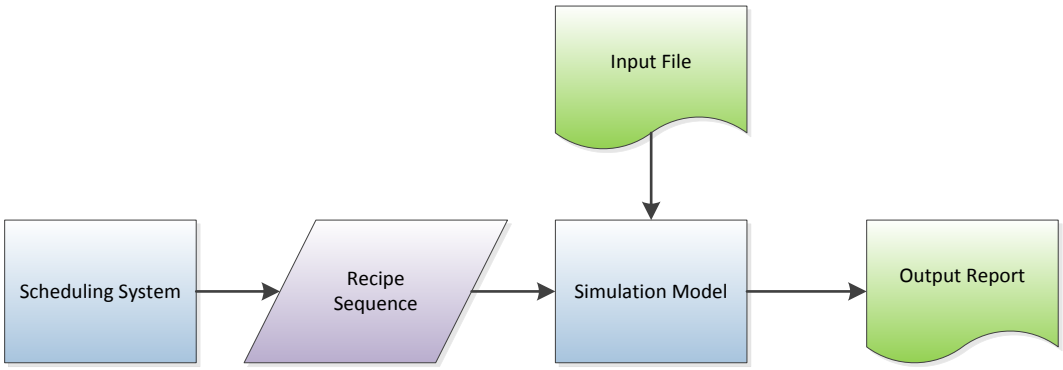

Figure 5.1: Approach flow.

5.2 Simulation Model

In order to develop an executable model that simulates the Implant tool in a correct manner, a systems architecture development methodology (SADM) was considered necessary. This methodology guaranteed the development of a system, an executable model in this case, that was responsive to customer's needs.

For this project we used Unified Modeling Language (UML) for the architecture viewpoint diagrams needed to create our executable model. Furthermore, we chose Colored Petri Nets (CPN) to develop our executable model because this tool supports UML and is greatly used in the cluster tool literature.

Colored Petri Net (CPN) is a graphical tool used to construct models and analyze system properties. CPNs allow for the creation of executable models that help describe system processes and capabilities, while validating behavior predictions. Furthermore, CPNs allow for the analysis of mathematical models of the physical systems themselves through their state space analysis capabilities.

In summary the reasons and benefits of using UML and CPN are as follows:

a) The existence of a methodology for converting UML class diagrams into CPN models
b) CPN's highly visual development that aids in the understanding of model operation
c) CPN's powerful analytic capabilities such as state space analysis
d) UML and CPN's compatibility with windows operating systems

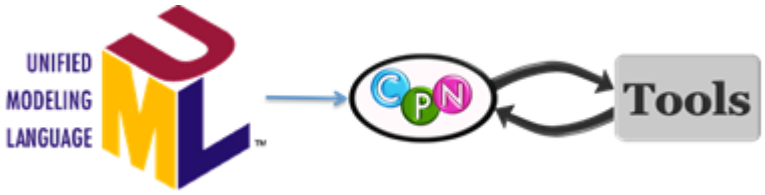

Figure 4.2: DODAF-CPN Transformation Process

5.3 Scheduling System

In order to improve the throughput of their implant tools, companies often schedule their recipe sequences to process lots in the shortest possible amount of time. This process is done with a scheduling system.

CTSG initially chose to perform a state space analysis to find the shortest recipe sequence for simulation in our implant model. In state space analysis, the simulation tool goes through every possible combination of recipes and stores them in different states, creating branches for all the possibilities. Then by running a search for the ending branches of the state space tree, the different states where all

the recipe lots have been processed can be searched until the sequence that gives the shortest processing time has been found.

One major issue with state space analysis is that it needs to go through every possible state for a sequence of recipe lots. As the sequence grows, the tree grows immensely, making the running process very slow. We found that the running time for state space analysis was reasonable for sequences up to 8 recipes long. For longer sequences, it could take hours for the state space tree to be created.

As a result, we looked for a faster tool to create the recipe sequences to test in our simulation tool. We chose to develop three types of scheduling algorithms in C. The first algorithm goes through all possible combinations of the recipe sequence, which is similar to state space analysis. However, this algorithm doesn't create a state space tree. As a result it is much faster than state space analysis. The second and third algorithms are versions of greedy algorithms.

A greedy algorithm is a mathematical process that works to find good solutions to difficult optimization problems in reasonable amounts of time. The process focuses on incrementally finding an approximate (good) solution by making myopic decisions, i.e., making an optimal decision at a given phase, ignoring the consequences of that decision on future decisions.

5.4 Approach Execution

Once a recipe sequence was obtained through the scheduling algorithms, the recipe sequence was tested in the Colored Petri Net model to check if it would be completed successfully or if the ion source would die before all the recipes in the sequence were processed. The results obtained with the three different algorithms were compared in a sample trade-off analysis of processing time and ion source life to demonstrate the usefulness of the tool.

5.5 Use Case

The following is the use case for the main success scenario of our implant simulation tool. The use case describes the steps followed when executing a simulation run.

5.5.1 Characteristic Information

The following defines information that pertains to this particular use case.

Goal In Context:	To simulate the implant process for a train (group) of lots
Scope:	Simulation Tool
Level:	Task
Pre-Condition:	Train of lots has arrived at the implant tool

Success End Condition:	Simulation has ended and a report is returned
Minimal Guarantees:	Simulation runs, but doesn't give an output report
Primary Actor:	User
Trigger Event:	User provides an input file

5.5.2 Main Success Scenario

This Scenario describes the steps that are taken from trigger event to goal completion when everything works without failure.

Step	Actor	Action Description
1	User	Provides an input file with recipe information and lot information
2	User	Provides a recipe sequence to simulate
3	User	Runs the simulation model
4	Model	Reads input file
5	Model	Processes lots and keeps track of ion source life
6	Model	Creates a output report with #of lots processed, #of lots unprocessed, ion source status, warnings if any, and #of ion source changes

6 Project Plan

6.1 Deliverables

The following table contains a list of the deliverables for this project and their due dates.

Table 5.1: Project Deliverables

Date	Deliverable
9/6/12	Problem Definition Presentation
9/13/12	Project Proposal Presentation
9/27/12	In-Progress Report 1 Document & Presentation
9/28/12	Monthly informal Progress Report for Sponsor
10/11/12	In-Progress Report 2 Document & Presentation
10/18/12	In-Progress Review Presentation
10/26/12	Monthly Informal Progress Report with Sponsor
11/1/12	Draft of Final Presentation
11/15/12	Draft of Final Presentation and Final Report
11/29/12	Dry Run of Final Presentation
11/29/12	Final Report Document
11/30/12	Monthly Informal Progress report with Sponsor
12/7/12	Final Presentation

6.2 Work Breakdown Structure

The following table contains the breakdown of the tasks required to complete this project.

Table 6.2: WBS

Outline Number	Task Name
1	**Cluster Tool Simulation Project**
1.1	**Initiation**
1.1.1	Project Team Kickoff Meeting
1.1.2	Define Problem and Goals
1.1.3	Conduct Preliminary Research
1.2	**Planning**
1.2.1	Allocate Tasks to Project Team
1.2.2	Write Proposal
1.2.3	*Deliverable 1 Submit Proposal to Sponsor*
1.2.4	*Deliverable 2: Submit Proposal to Dr. Hoffman*
1.2.5	Project Sponsor Reviews Proposal
1.2.6	Project Proposal Signed/Approved
1.2.7	Research Architecture Framework, Modeling and Analysis tools
1.3	**Project Management**
1.3.1	Create Schedule
1.3.2	Allocate Tasks to Project Team
1.3.3	Create and Update Project Plan
1.3.4	**Internal Project Status Meetings**
1.3.5	**External Project Status Meetings**
1.3.6	**In- Progress Reports**
1.4	**Execution**
1.4.1	Project Kickoff Meeting
1.4.2	Verify & Validate User Requirements
1.4.3	**Develop Simulation Model**
1.4.3.1	Design Architecture Viewpoints
1.4.3.2	Create CPN Model
1.4.3.3	Update Architecture Viewpoints
1.4.3.4	Update CPN Model
1.4.4	**Develop Scheduling System**
1.4.4.1	Perform State Space Analysis
1.4.4.2	Develop Scheduling Algorithms
1.4.4.3	Test and debug Algorithms
1.4.4.4	Perform Validation Testing

1.4.4.5	Perform Scheduling Analysis
1.4.4.6	Provide Recommendations
1.4.5	Create and Update Project Website
1.4.6	*Deliverable 3: Submit Draft of Final Report*
1.4.7	Update Final Report & Presentation
1.4.8	*Deliverable 4: Give Dry-Run of Final Presentation*
1.5	**Closeout**
1.5.1	*Deliverable 5: Submit Final Report to Dr. Hoffman*
1.5.2	*Deliverable 6: Submit Final Report to Sponsor*
1.5.3	*Deliverable 7: Give Final Presentation*

6.3 Schedule

The following figure contains our project schedule.

	Outline Number	Task Name	Start	Finish
1	1	**Cluster Tool Simulation Project**	**Thu 8/30/12**	**Fri 12/7/12**
2	1.1	**Initiation**	**Thu 8/30/12**	**Wed 9/5/12**
5	1.2	**Planning**	**Thu 9/6/12**	**Tue 9/18/12**
4	1.3	**Project Management**	**Thu 9/6/12**	**Thu 12/6/12**
14	1.4	**Execution**	**Tue 9/18/12**	**Fri 11/30/12**
15	1.4.1	Project Kickoff Meeting	Mon 9/17/12	Mon 9/17/12
16	1.4.2	Verify & Validate User Requirements	Mon 9/17/12	Thu 9/20/12
17	**1.4.3**	**Develop Simulation Model**	**Mon 9/17/12**	**Fri 11/9/12**
18	1.4.3.1	Design Architecture Viewpoints	Mon 9/17/12	Fri 10/5/12
19	1.4.3.2	Create CPN Model	Fri 10/5/12	Tue 10/23/12
20	1.4.3.3	Update Architecture Viewpoints	Mon 10/8/12	Fri 10/26/12
21	1.4.3.4	Update CPN Model	Wed 10/24/12	Fri 11/9/12
22	**1.4.4**	**Develop Scheduling System**	**Mon 10/8/12**	**Tue 11/27/12**
23	1.4.4.1	Perform State Space Analysis	Mon 10/8/12	Tue 10/16/12
24	1.4.4.2	Develop Scheduling Algorithms	Tue 10/16/12	Tue 10/30/12
25	1.4.4.3	Test and debug Algorithms	Tue 10/30/12	Fri 11/9/12
26	1.4.4.4	Perform Validation Testing	Mon 11/12/1	Fri 11/16/12
27	1.4.4.5	Perform Scheduling Analysis	Mon 11/19/1	Fri 11/23/12
28	1.4.4.6	Provide Recommendations	Mon 11/26/1	Tue 11/27/12
29	1.4.5	Create and Update Project Website	Mon 10/29/12	Fri 11/30/12
30	*1.4.6*	*Deliverable 3: Submit Draft of Final Report*	*Thu 11/1/12*	*Thu 11/1/12*
31	1.4.7	Update Final Report & Presentation	Mon 11/26/12	Wed 11/28/12
32	1.4.8	*Deliverable 4: Give Dry-Run of Final Presentation*	Thu 11/29/12	Thu 11/29/12
33	**1.5**	**Closeout**	**Thu 11/29/12**	**Fri 12/7/12**
34	*1.5.1*	*Deliverable 5: Submit Final Report to Dr. Hoffman*	*Thu 11/29/12*	*Thu 11/29/12*
35	*1.5.2*	*Deliverable 6: Submit Final Report to Sponsor*	*Fri 11/30/12*	*Fri 11/30/12*
36	*1.5.3*	*Deliverable 7: Give Final Presentation*	*Fri 12/7/12*	*Fri 12/7/12*

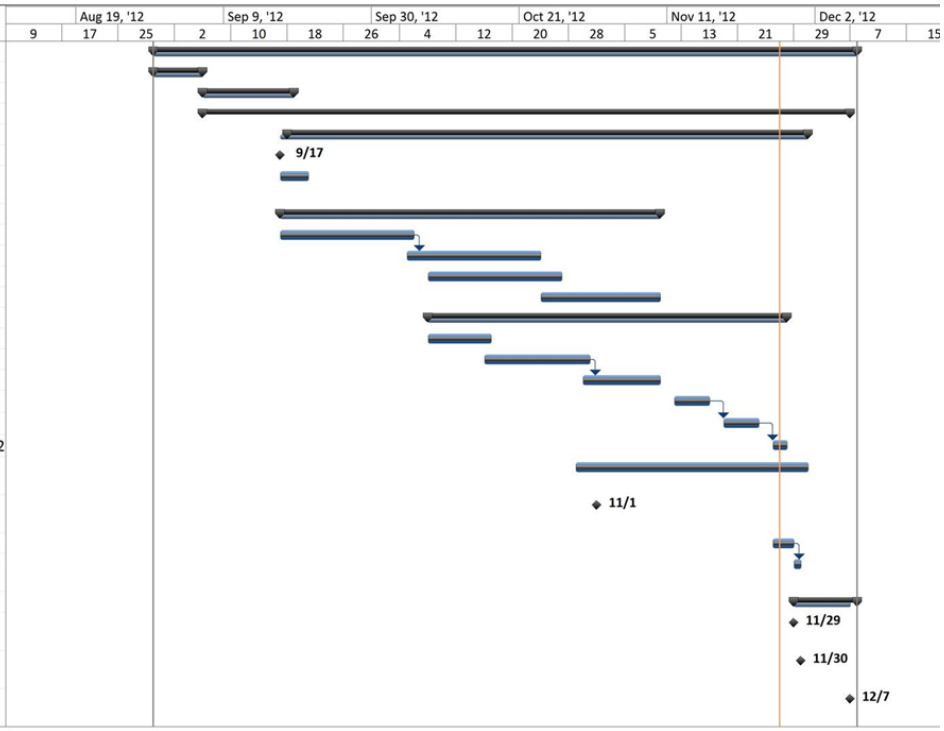

Figure 6.3: Project Schedule

7 Design

7.1 Simulation Model

Unified Modeling Language (UML) was used to develop three key view point diagrams for the development of the simulation model. The three types of object oriented diagrams created were a Class Diagram, an Activity Diagram, and a State Transition Diagram.

The first step in developing the simulation tool was to develop a Class Diagram (Figure 6.1a). The Class Diagram helps to identify the components that the simulation tool should have. Additionally, it gives an idea of the basic relationships between components.

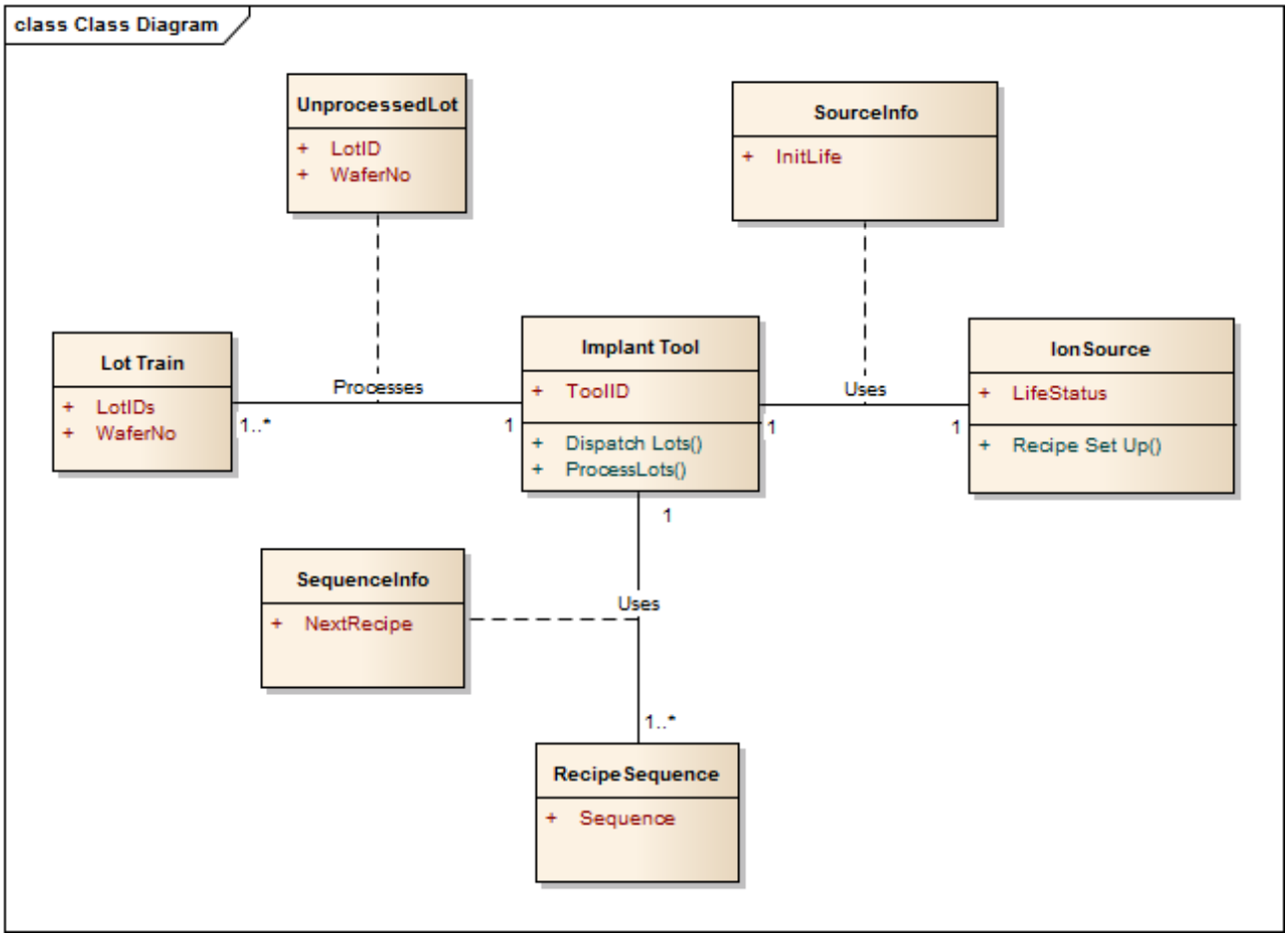

Figure 6.1a: Class Diagram

The classes for the simulation tool are: Lot Train, Implant Tool, Ion Source, and Recipe Sequence. “Lot Train” only holds information; therefore, it does not have any operations. It contains information on the ID number for the lots and the number of wafers in each lot. Similarly, “Recipe Sequence” does not have any operation either.

This class contains information on the lot processing sequence. The class “Implant Tool” holds information and performs operations. It contains information about the tool ID and performs two operations: it dispatches and processes lots. Lastly, the class “Ion Source” holds information and performs one operation. This class contains information on the status of the source, and it sets up the recipes for the implantation.

All of these classes are related by association classes that act as packets of information that are passed among components. This is a basic form of modeling how the components interact with one another. We can interpret the class diagram as follows: classes “Lot Train,” “Ion Source,” and “Recipe Sequence” pass or send the packets “Unprocessed Lot,” “Sequence Info,” and Source Info” to class “Implant Tool.” This information is used by class “Implant Tool” to process the next lot and dispatch it.

The next step in the development of the CPN simulation tool was to develop an Activity Diagram. This was done in order to model the behavior that the CPN simulation model should follow. The Activity Diagram is shown on figure 6.1c.

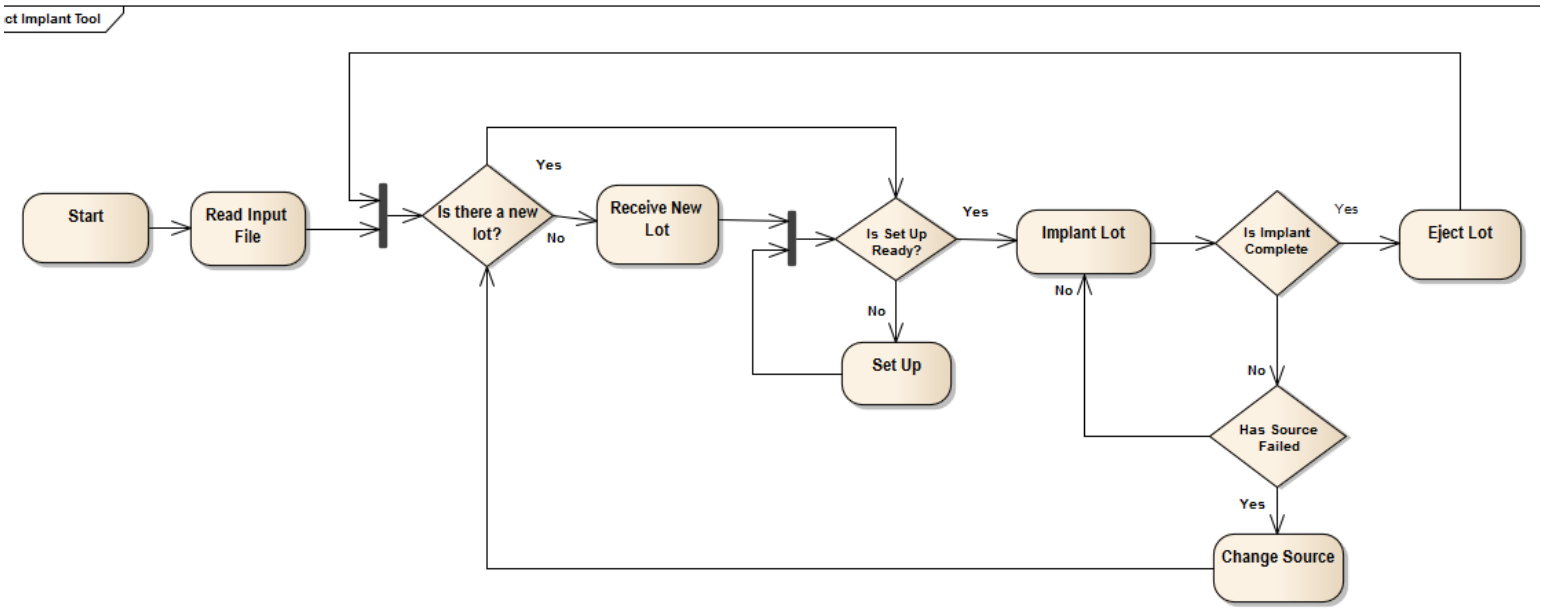

Figure 6.1c: Activity Diagram

The first activity that the simulation tool performs is to read the input file. Then it checks to see if it already has an unprocessed lot that is ready to be processed. If the tool has an unprocessed lot, it proceeds to the next step. If it does not, the tool gets a new unprocessed lot. Next, the tool checks to see if the recipe set-up is ready. If that is the case, the tool starts the implantation. If the set-up has not been done, the tool initiates the set-up. After implantation, the tool checks if the implantation has been completed. If it has, the tool dispatches the lot. If the implantation was not completed, the tool checks if there was a source failure. After the tool has checked for a source failure, it can go back to implantation, or it can perform a source change, depending on the outcome of the check. This diagram helps identify the behavior of the simulation tool.

Finally, a State Transition Diagram was developed in order to establish the main capabilities for the CPN model. The State Transition Diagram is depicted on Figure 6.1b.

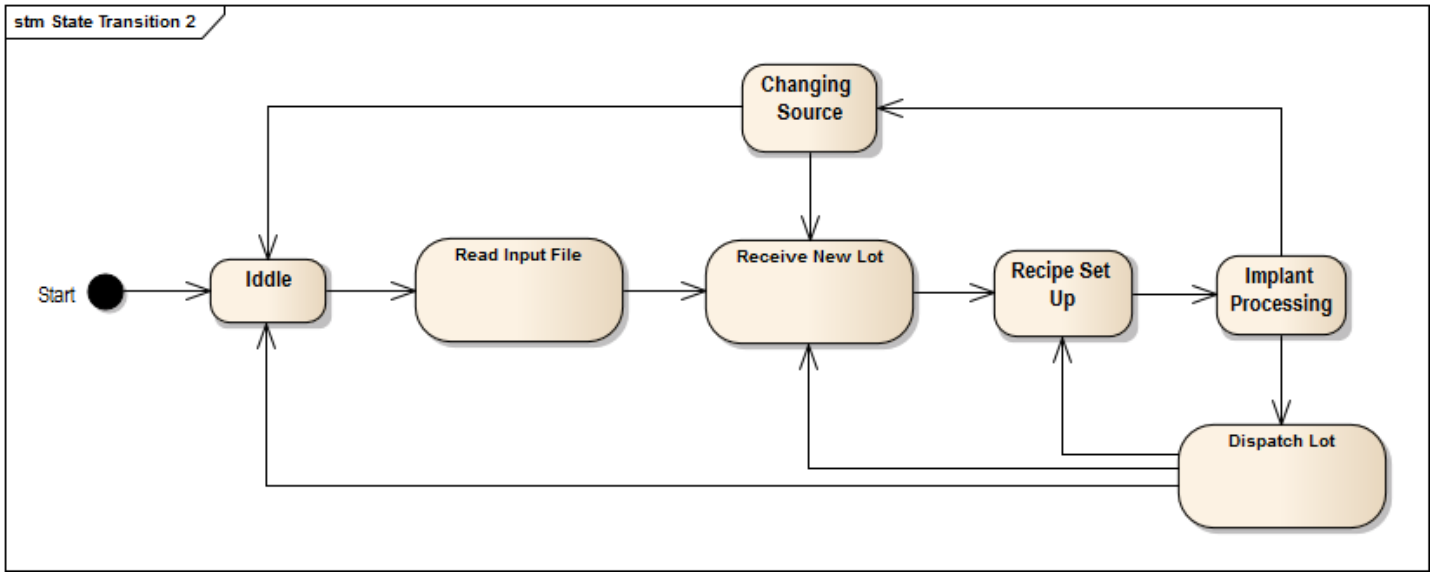

Figure 6.1b: State Transition Diagram

The State Transition Diagram depicts the states that the implant tool can be in during operations. The first state is the idle state. This state occurs when the tool is ON, but it is not processing any lots. From idle the tool can go to the state Read Input File. This occurs when the tool is populating all its variables from a user defined input text file. From here, the tool goes to the Receive New Lot state. The tool is in this state when it is waiting for the next lot for implantation. Next, the tool goes into the Recipe Set-Up state. During this state, the Ion Source is generating the desired plasma, specified by the lot recipe. Additionally, the tool cannot do anything else when it is in the set-up state. Once the set-up is ready, the tool goes into the Implantation state. This is the state the tool is in when it is processing a lot. After implantation, the tool can go to Dispatch Lot or to Change Source (maintenance), depending on the state of the source. When the tool is in the Changing Source state, it can go to the Idle state if there are no more lots to be processed or to the Receive New Lot state, where the whole process is repeated. Finally, when the tool is in the Dispatch Lot state, it can go to the Idle state if there are no more lots to be processed, the Receive New Lot state, or Recipe Set Up state.

The three diagrams discussed above, when put together, provide a complete picture of how the CPN simulation tool should operate in order to emulate operations of a real Implant Tool. The model we created by combining the Class Diagram, Activity Diagram, State Transition Diagram with Colored Petri Nets is shown in figures 6.1d.

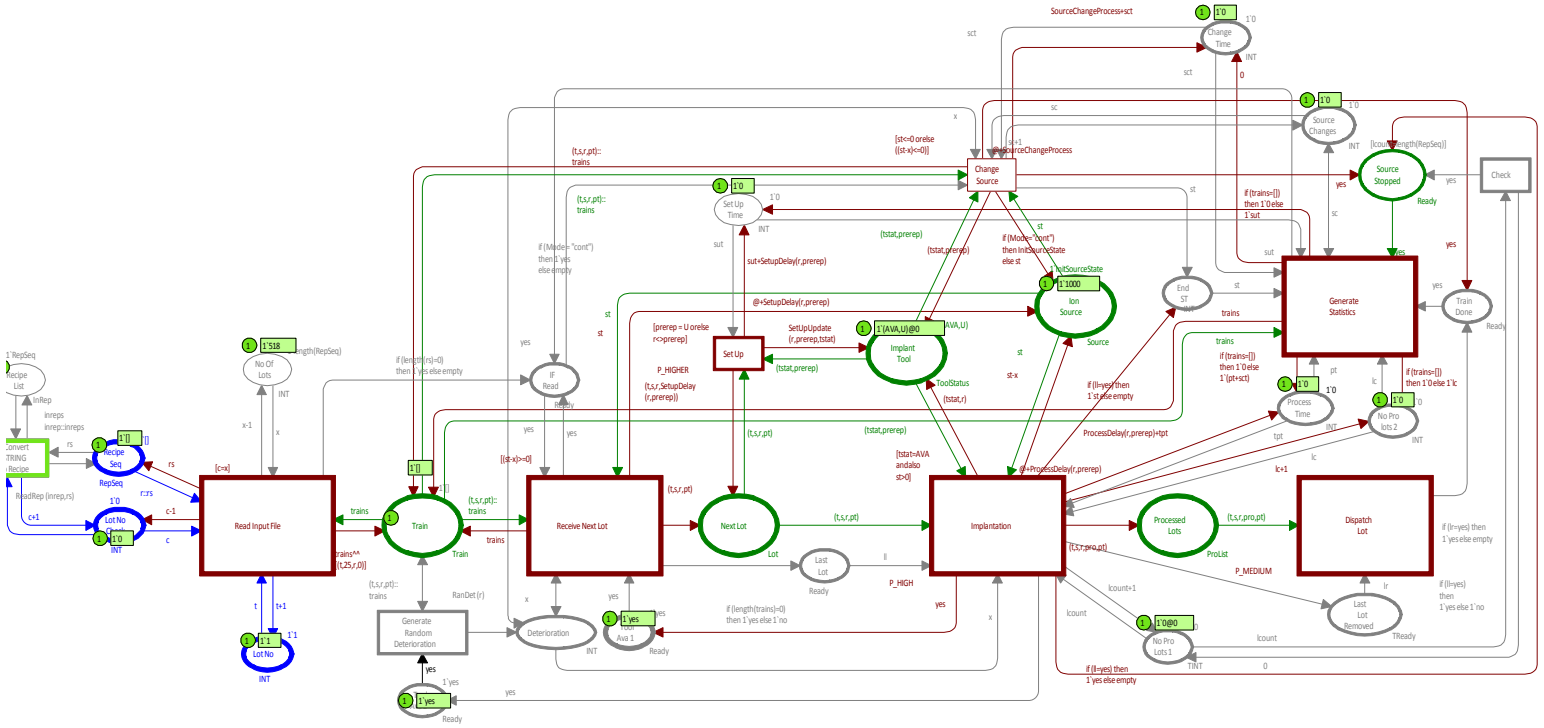

Figure 6.1d: CPN Implant Simulation Model

The CPN model shown on Figure 6.1d contains the classes, activities, and states from the three UML diagrams discussed above.

7.2 Scheduling System

Originally, Micron Technologies, our previous sponsor was going to provide us with scheduling algorithms to test in our tool. However, due to the proprietary nature of this information, and as a result the change of sponsor, we had to develop our own algorithms to test our tool. It is important to note that the development of the scheduling algorithms was just for the demonstration of our simulation tool. The algorithms were not the main focus of this project.

CTSG used C to develop the scheduling algorithms. We developed three different scheduling algorithms to find recipe sequences with different processing times. The purpose of the three scheduling algorithms is to compare them after simulation to demonstrate how the simulation tool can be used to evaluate scheduling techniques based on processing time and ion source deterioration.

The first scheduling algorithm searches for the different kinds of recipes that need to be organized (without repetitions). Then, it organizes these recipes by going through all their permutations, until the sequence with the shortest processing time has been found. Once the shortest permutation has been found, the algorithm concatenates the repetitions of each recipe at the end of their original recipe, so that the bigger sequence is also the shortest permutation to process in the implant tool. The way this algorithm works is by taking a big problem, breaking it down into a smaller problem that is easier to solve, and then applying the solution of the smaller problem to the big problem. This way, the algorithm can run much faster than if it had to go through all the permutations of the longer recipe sequence, which could take several hours to complete. The flow chart for the first scheduling algorithm is shown on figure 7.2a

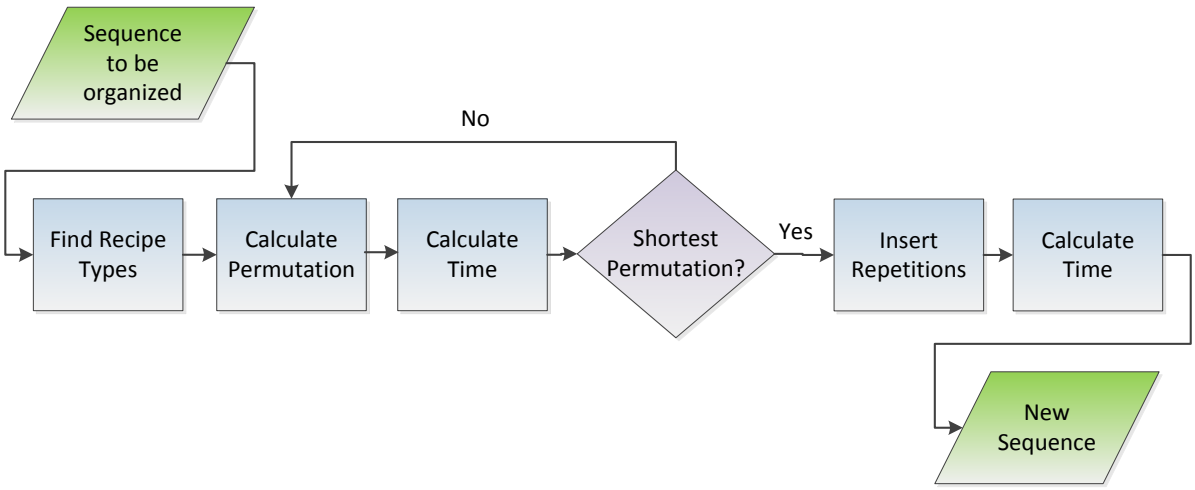

Figure 7.2a: Algorithm 1

Our second scheduling algorithm is a greedy algorithm that continuously picks the recipe with the shortest processing time (with set-up times included) until all recipes have been picked. This way, the sequence is organized from smallest to largest. The flow chart for this algorithm is shown in figure 7.2b.

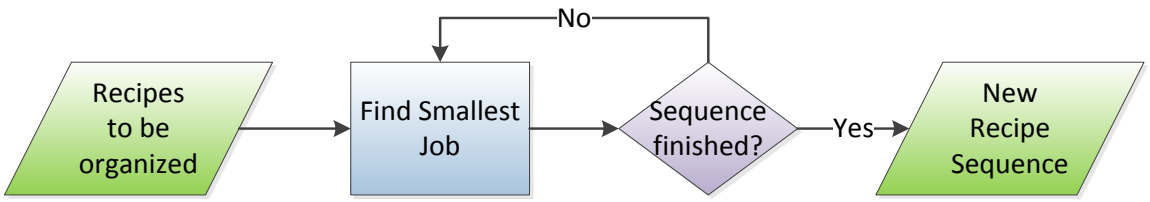

Figure 7.2b: Algorithm 2

The third scheduling algorithm is a greedy algorithm that picks the longest jobs that need to be processed and organizes the sequence from largest to smallest. The flow chart for this algorithm is shown in figure 7.2c.

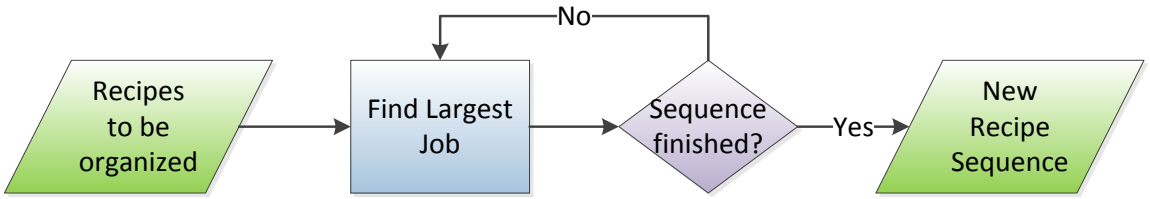

Figure 7.2c: Algorithm 3

The algorithms were created with 5 different recipes types (A,B,C,D,E), each of which has processing times associated with it. Furthermore, there is a set-up time for the implant tool in between different recipes, which is the time that the implant tool spends on preparing the gases, temperature, pressure, etc for recipe changes. While some recipes have symmetrical set-up times i.e., it takes the same amount of time to go from recipe A to recipe B or recipe B to recipe A (A->B = B->A), not all times are symmetric. Therefore, we allowed for asymmetric set-up times in our scheduling algorithms. This lets us explore more recipe combinations that result in different deterioration amounts on the life of the ion source.

8 Testing

8.1 Assumptions and Limitations

In order to simplify the model to meet time constraints, assumptions were made regarding the arrival of lots at the implant tool, the delays between recipes, and the deterioration effects on the ion source.

The first assumption relates to the availability time of lots (lot train) that need processing. In reality, lots become available at different times. Semiconductor processing companies, such as Micron, use a scheduling system that knows when the unprocessed lots will be available, and produces a schedule for the Implant tool based on that. For the analysis CTSG is doing to demonstrate the usefulness of the tool, we are assuming that all unprocessed lots are available at time zero, thereby allowing us to generate all possible processing order scenarios with our scheduling algorithms.

The second assumption relates to the time delay between lots. We assumed each lot is processed immediately after the previous lot has been processed and that any necessary ion source set-up has occurred; no delays are allowed.

The third assumption relates to the ion source deterioration. We assumed, for demonstration purposes, that the deterioration effect of each recipe is normally distributed with some arbitrary mean and variance as we don't have this data. However, users may change the deterioration effects to test scheduling techniques with the effects of their own recipes. This could be done through the use of random variables or deterministic numbers obtain from data.

8.2 Validation Testing

In order to validate that our simulation model correctly models an implant tool, we asked the Systems Architectures Lab (SAL) to run a set of recipe sequences with the same parameters we used in our simulation model. Since, we don't have an implant tool to test our model against; our sponsor's model is the closest thing to an implant tool that we can use since it has been proved to be an accurate model by Micron Technology, our previous client.

It should be noted that the validation testing was done without the source deterioration effects because our effects were chosen arbitrarily for demonstration purposes. The effects need to be known by the user in order to use them in the simulation model. Furthermore, testing the results would require us to run the recipes in a real implant tool, which is beyond the scope of this project.

Note: The recipe times, set-up times, and deterioration effects for testing and for the trade-off analysis were chosen arbitrarily. However, the client can input their own data for these parameters to simulate their own implant processes.

8.3 Validation Results

The validation testing showed that both SAL's implant model and our simulation model produce the same results, with the same input parameters. Unfortunately, due to confidentiality issues, this data cannot be shared. However, the results prove that the simulation model correctly resembles the behavior of an implant tool. The results can be verified with our sponsor.

9 Analysis

9.1 Scheduling Trade-off Analysis

The purpose of this analysis is to show the benefits and the types of analysis that can be performed with our simulation model; it is a proof-of-concept (POC). In this analysis a trade-off study was performed in order to show that the implant tool can be used to evaluate recipe sequences based on total processing time and ion source deterioration.

For this analysis, CTSG evaluated the three basic scheduling algorithms presented in section 7.2. These algorithms provided sequences with different total processing times, which were then compared in the simulation tool.

9.2 Results

Thirty sequences were created for the three scheduling algorithms to organize. Each sequence contained a number of unprocessed lots associated with particular recipes (A, B, C, D, or E).

Table 9.2.1, on the next page, contains the sequence number along with the number of lots each sequence contains. Additionally, it contains the original sequence, which was obtained by using a random selector, and the organized sequences obtained through the three scheduling algorithms: shortest greedy and longest greedy, and the permutation algorithm.

Table 9.2.1: Recipe Sequences

Seq #	# of lots	Original Sequence	Shortest Greedy Algorithm	Longest Greedy Algorithm	Permutation Algorithm
1	122	{C, C, C, B, C, A, A, D, B, B, C, D, D, A, E, E, C, B, B, D, E, B, D, C, B, D, C, B, C, B, A, B, C, D, C, C, E, E, A, A, B, D, A, D, E, A, E, A, B, A, E, B, D, D, B, A, E, A, D, A, A, E, B, E, E, C, D, C, D, C, A, B, B, C, D, B, B, E, B, D, E, C, E, B, B, B, A, A, E, C, A, C, A, E, B, E, A, E, D, C, A, B, E, B, E, C, C, B, D, D, C,D, B, B, B, E, D, B, A, B, B, B}	{E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, C, C, C, C, C, , C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D}	{D, A, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B,C, B, C, B, C, B, C, B, C, B, C, B, C, B, A, B, B, B, B, B, B, B, B, B, B, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E}	{A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B,B, B, B, B, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E}
2	102	{B, D, D, B, B, D, A, A, D, E, C, C, E, E, C, E, D, D, E, A, A, E, E, B, D, E, E, E, A, E, B, B, A, E, E, E, D, C, C, C, B, A, B, B, B, B, B, B, E, E, D, D, A, B, A, D, B, D, E, D, C, C, B, D, D, E, A, A, C, E, E, B, A, B, A, A, E, A, D, D, A, B, E, B, B, E, D, D, B, D, B, B, C, B, A, E, D, D, D, E, E, B}	{E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, C, C, C, C, C, C, C, C, C, C, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D}	{D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, B, C, D, B, C, D, B, C, D, B, C, D, B, C, D, B, C, B, C, B, C, B, C, B, C, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B,B, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E}	{A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, A, C, C, C, C, C, C, C, C, C, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D,D, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E}
3	100	{A, C, A, C, A, E, E, C, B, B, B, D, B, E, E, D, E, E, B, E, D, E, E, B, E, E, D, E, B, C, B, D, C, E, B, E, B, E, D, B, C, E, E, D, D, D, C, C, E, A, E, C, D, D, E, C, C, B, D, C, D, E, D, C, E, B, E, D, C, B, E, E, B, D, A, E, B, E, A, E, D, A, A, E, D, A, D, A, A, B, E, D, A, A, C, B, E, A, C, D}	{E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, A, A, A, A, A, A, A, A, A, A, A, A, A, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D}	{D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, A, D, B, C, D, B, C, D, B, C, D, B, C, D, B, C, D, B, C, D, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B, C, B, B, E, E, E, E, E, E,E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E}	{A, A, A, A, A, A, A, A, A, A, A, A, A, A, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, C, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, E, E, E, E, E, E,E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E, E}

Table 9.2.1 contains three sample sequences that were reorganized by the different scheduling algorithms. For the full set of sequences, see the embedded spreadsheet from Appendix D.

The input data used in the input file for the simulation model is presented in tables 9.2.2 – 9.2.5 below.

Table 9.2.2: Process Times

Recipe	Time (min)
A	44
B	63
C	11
D	27
E	10

Table 9.2.2 contains the processing time for each recipe.

Table 9.2.3: Set-up times in minutes

i\j	A	B	C	D	E
A	0	10	12	13	11
B	11	0	14	15	12
C	14	12	0	11	15
D	9	13	10	0	10
E	15	9	12	11	0

Table 9.2.3 contains the set-up times between recipes. This table is read by looking at the row for the old recipe (i), and then the column for the new recipe (j). For example, the set-up time of going from recipe A to recipe A is 0, and the set-up time of going from recipe A to recipe B is 10 minutes.

Table 9.2.4: Deterioration Data

Recipe	Mean	Variance
A	10	3
B	30	9
C	15	4
D	30	8
E	20	6

The deterioration data (mean and variance) used to model the deterioration effects of each recipe as a normal distribution is presented in table 9.2.4.

Table 9.2.5: Source Data

Initial Source State	Source Change time (min)
1000	180

Table 9.2.5 contains the data related to the ion source: the initial state of a new source that has never been used before, and the amount of time it takes to change the ion source when it fails.

Each recipe sequence was run multiple times to sample the life of the ion source, which is affected by random deterioration. Then, the average of the source life at the end of a train run (group of lots) was calculated. The processing time for each sequence of recipes, the average remaining source life, and the average number of source changes are presented in table 9.2.6 on the next page.

Table 9.2.6: Simulation Results from the sequence runs.

Shortest Greedy			Longest Greedy			Permutation Algorithm		
Total Processing Time (min)	Remaining Source Life	Source Changes	Total Processing Time (min)	Remaining Source Life	Source Changes	Total Processing Time (min)	Remaining Source Life	Souce Changes
4564	122	2	5600	122	2	4559	122	2
3791	102	2	4456	102	0	3786	102	2
3217	100	2	3984	100	2	3212	100	2
3622	103	2	4578	103	2	3617	103	2
3761	111	2	4742	111	2	3759	111	2
3526	108	2	4440	108	2	3521	108	2
3617	109	0	4655	109	0	3612	109	2
3743	108	2	4712	108	2	3738	108	2
4096	112	2	5029	112	2	4091	112	2
4162	108	2	4983	108	2	4157	108	2
5406	156	3	6886	156	3	5401	156	3
5803	164	3	7262	164	3	5798	164	3
6599	172	3	8094	172	3	6594	172	3
5995	167	3	7680	167	3	5990	167	3
5293	155	3	6404	155	3	5288	155	3
5653	155	3	7004	155	3	5648	155	3
5919	178	3	7525	178	3	5914	178	3
5469	158	3	6888	158	3	5464	158	3
6210	184	3	7949	184	3	6205	184	3
5557	152	3	7088	152	3	5552	152	3
7926	227	4	10063	227	4	7921	227	4
6381	207	4	7943	207	4	6376	207	4
8242	229	4	10360	229	4	8237	229	4
6825	201	4	8685	201	4	6820	201	4
17706	518	10	22550	518	10	17701	518	10
4565	120	2	5784	120	2	4561	120	2
4312	110	2	5452	110	2	4308	110	2
5201	123	2	6028	123	2	5198	123	2
3743	117	2	4784	117	2	3741	117	2
3392	77	1	3836	77	1	3387	77	1

Figure 9.2.1, presented below, shows the average total process time versus the remaining source life for each recipe sequence, for the three scheduling algorithms. In this figure, each data point is represented by a diamonds, squares or triangles, which correspond to the shortest greedy algorithm, longest greedy algorithm, and the permutation algorithm, respectively. More specifically, each data point represents a sequence that was run by following one of the three scheduling algorithms. Furthermore, each data point shows how fast a given sequence was completed and the remaining source life after the processing was completed. Note that only 10 sequences were plotted in the figure below to make the plot easier to read. For a more detail look, please refer to the excel worksheet in appendix D.

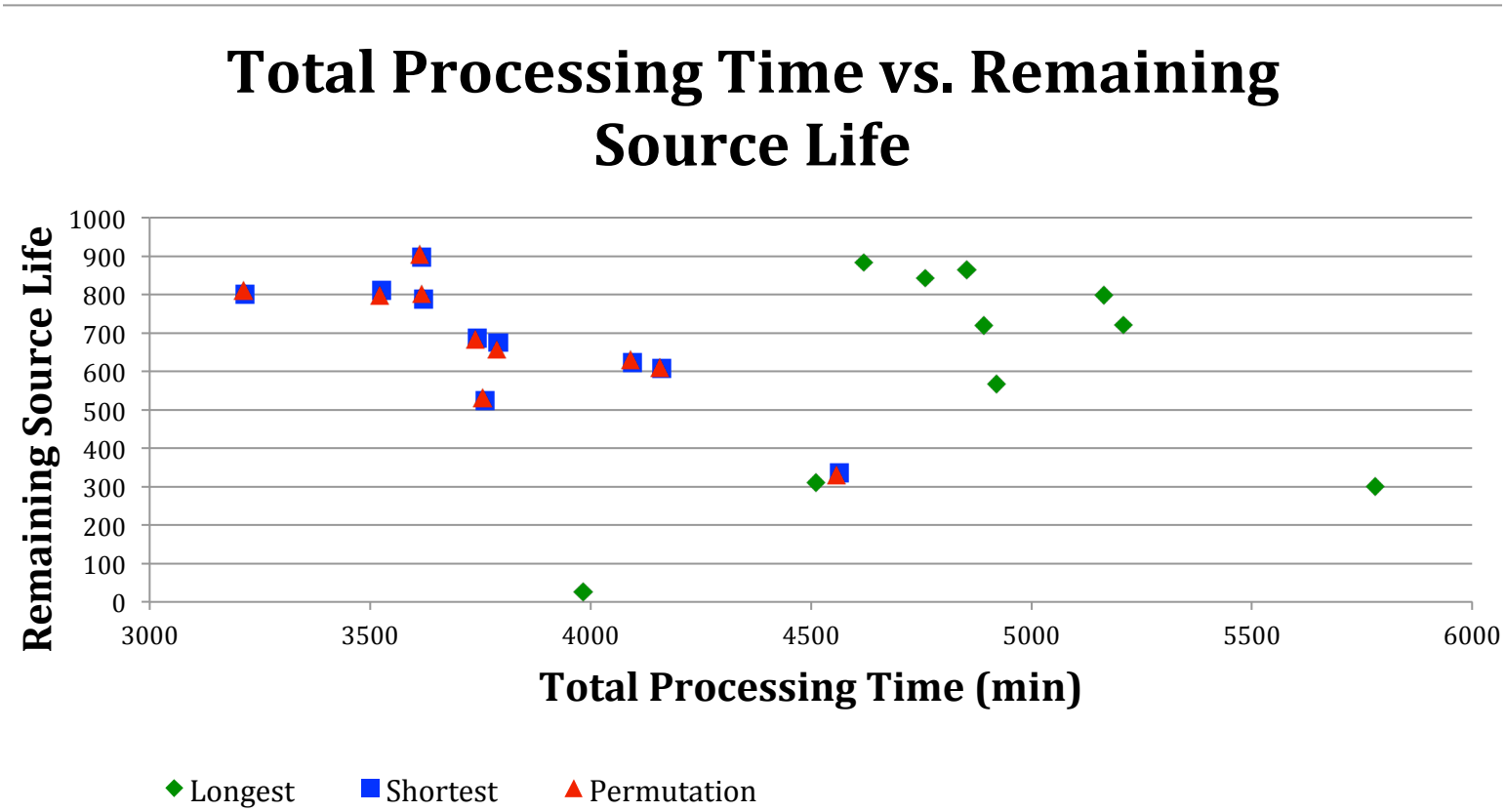

Figure 9.2.1: Processing Time vs. Source Life

In order to take a closer look at the data, we have presented the results for the total processing time and the remaining life of the ion source in separate plots. Figure 9.2.2 shows the average total processing time for ten of the sequences we ran. Here, it is evident that the longest greedy sequences have the slowest processing times. Additionally, we can see that the shortest greedy sequences and the permutation sequences are very close to each other. However, the permutation sequence, on average, are faster by five minutes.

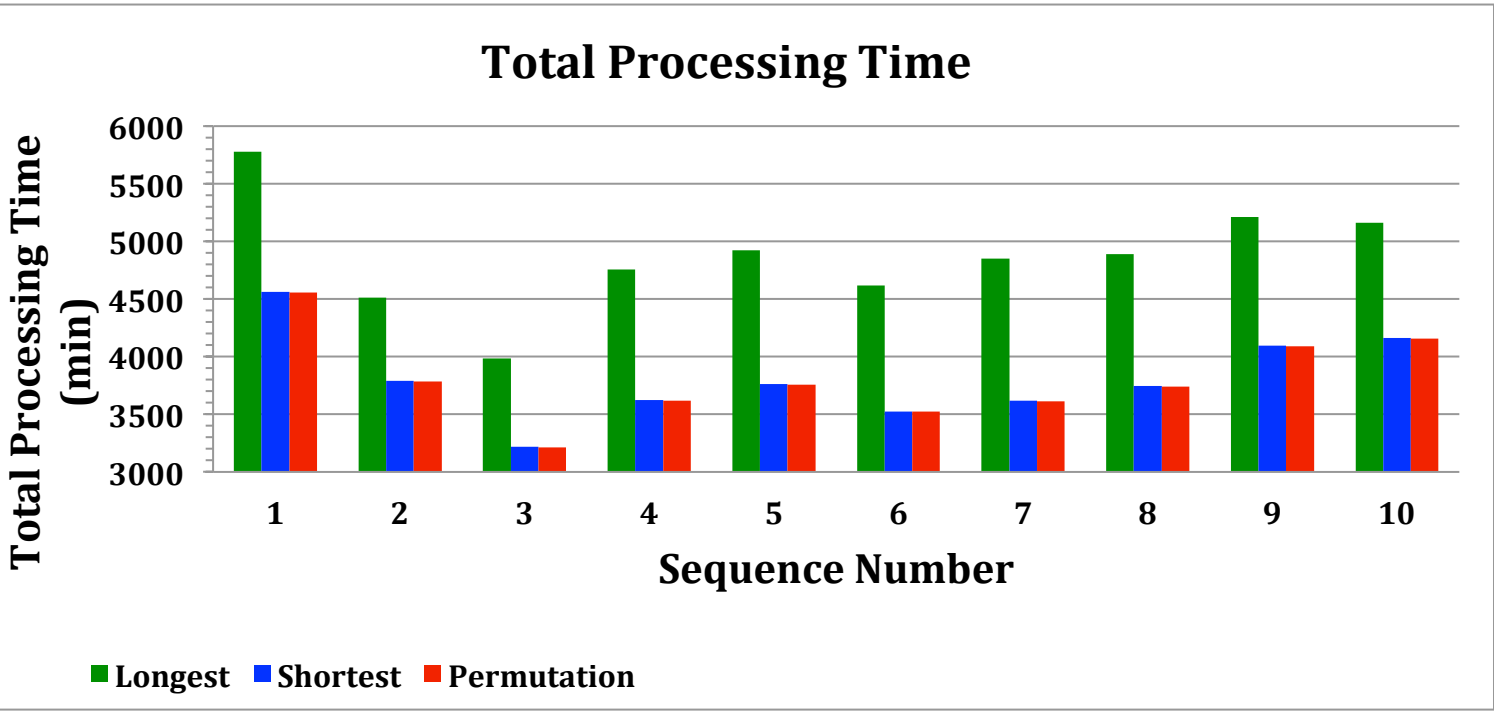

Figure 9.2.2: Total Processing Time

Figure 9.2.3 shows the average remaining life for the last ion source in the implant tool at the end of the simulation run for the same ten sequences compared above. Here, in some cases it seems that the longest greedy algorithm has the best ion source remaining life. However, the higher remaining life is due to additional ion source changes near the end of the simulation, that is, when the ion source was changed close to the completion of the simulation run. For a closer comparison of the data, we calculated the average of all the runs, which are presented on table 9.2.7.

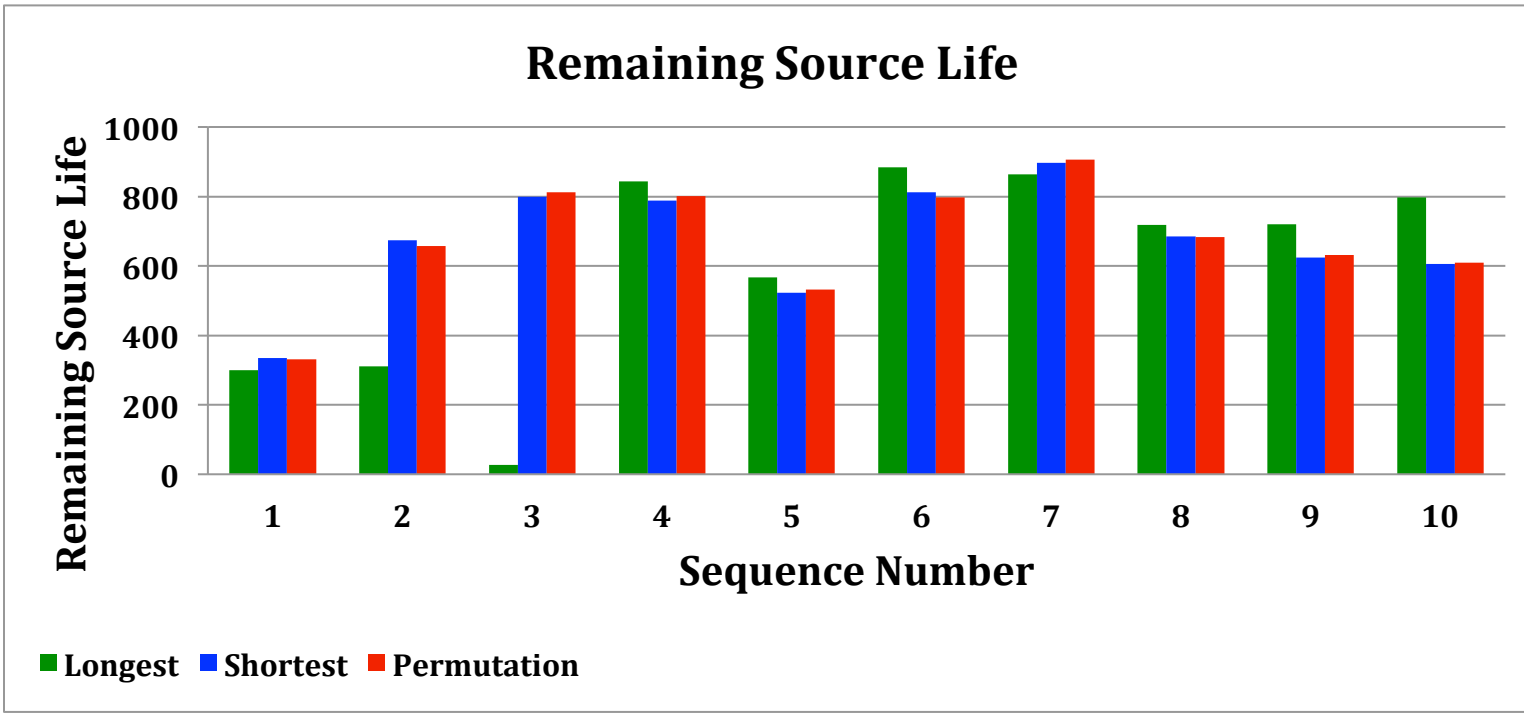

Figure 9.2.3: Remaining Source Life

Table 9.2.7: Average Results

Algorithm	Average Processing Time (min)	Average Remaining Source Life	Average Source Changes
Longest Greedy	4869.1	602.92	2.83
Shortest Greedy	3809.9	674.45	2
Permutation	3804.9	676.08	2

Table 9.2.7 contains the average total processing time as well as the average remaining source life for the ten sequences compared earlier Here it is clear that the longest greedy sequences have the poorest performance. They have the longest total processing times, as well as the lowest remaining source life due to the additional ion source changes. Furthermore, we can see that the results for the shortest greedy and the permutation sequences were very close to each other. This is because both schedules try to minimize the set-up times in between recipes. As a result, the permutation sequences, which have the smallest total set-up times, have the best results both for fastest total processing time, as well as for the highest average ion source remaining life.

From these results we can see that the longest greedy schedules had the greatest effect on the deterioration of the ion source because of their long set-up times, while the other two algorithms helped decrease ion source deterioration by decreasing the set-up times.

It must be emphasized that this is only a proof of concept analysis; therefore, we are not making any recommendations as to which scheduling algorithm to use. Instead, we are providing a tool that will return data to help planning or scheduling engineers improve their schedules.

9.3 Recommendations

The simulation model that CTSG has developed is a powerful analysis tool that can be used for many purposes. Our model can be used to test different scheduling techniques or methodologies before implementation. This will allow scheduling or planning engineers to show the benefits of a proposed technique to management without the potential dangers of experimental testing.

The tool can also be used to identify whether the scheduling system that is currently in use needs improvements. The areas where improvements are needed can be identified from analysis of simulation results.

Additionally, this tool can be used to study any scheduling process that is affected by constraints. One good example is process chambers that need to be maintained after a certain period of time. The tool would provide insight into how to schedule the work to a process chamber and when to perform the necessary maintenance work.

9.4 Future Work

CTSG's simulation tool can be modified to study more than one tool or machine concurrently. For example, the tool can be expanded to study the scheduling of a group of implant tools (known as a workstation). In the case of implant tools, this type of study would prove beneficial by showing the time savings from a particular scheduling logic that could translate into cost savings. It can also be modified to study other kinds of cluster tools or to analyze multiple failures.

As a matter of possible future expansions, it may be possible to include deterioration and time into a combined metric and to develop more advanced scheduling algorithms to test in the simulation tool. For example, the Permutation algorithm may be modified by splitting repetitions based on deterioration metrics, in order to further extend the ion source's life. Although, this will increase processing time as was indicated by our sample analysis. The development and

analysis of this modified Permutation algorithm with the tool we have developed could be a possible capstone project for future classes.

10 References

Zuberek, W.M.; , "Cluster tools with chamber revisiting-modeling and analysis using timed Petri nets," *Semiconductor Manufacturing, IEEE Transactions on* 17.3 (2004): 333-334. IEEE Xplore. Web.

Naiqi Wu; Chengbin Chu; Feng Chu; Meng Chu Zhou; , "A Petri Net Method for Schedulability and Scheduling Problems in Single-Arm Cluster Tools With Wafer Residency Time Constraints," *Semiconductor Manufacturing, IEEE Transactions on* , 21.2 (2008): 224-237. IEEE Xplore. Web.

H.-Y. Lee and T.-E. Lee "Scheduling single-armed cluster tools with reentrant wafer flows", *IEEE Trans. Semicond. Manuf.*, 19.2 (2006): 226-240. IEEE Xplore. Web.

Dimmler, M.A. "Using simulation and genetic algorithms to improve cluster tool performance," *Simulation Conference Proceedings,* 1 (1999):875-879. IEEE Xplore. Web.

Poolsup, S.; Deshpande, S. "Cluster tool simulation assists the system design," *Simulation Conference, 2000,* 2 (2000): 1443-1448. IEEE Xplore. Web.

Srinivasan, R.S. "Modeling and performance analysis of cluster tools using Petri nets" *Semiconductor Manufacturing, IEEE Transactions on* , 11 (1998): 394-403. IEEE Xplore. Web.

Han-Pang Huang; Che-Lung Wang; "The modeling and control of the cluster tool in semiconductor fabrication," *IEEE International Conference on Robotics and Automation. Proceedings 2001 ICRA. IEEE,* 2 (2001): 1826- 1831. IEEE Xplore. Web.