SYSEN 5100 Final Report

CUSD Mobility - TCAT Lansing Routing Problem

Special Thanks:-

- **Dr. David Schneider**
- **Sirietta Simoncini**
- **Dr. Wenqi Yi**
- **CUSD Mobility Team**
- **TCAT**

By:-
Bakulesh Singh (bs774)
Systems Engineering, M.Eng'18

Table of Contents

Introduction

TCAT (Tompkins County Area Transit) has been observing low ridership levels in the Lansing Town area for the past few years. They approached the CUSD Mobility team to help them understand the reason for this and improve service based on our findings. They essentially wanted the team to redesign their system of operation for Lansing. This is exactly the kind of problem for which taking a step back and taking a systems view of the whole situation proves much more effective than just attacking apparent pain points to solve the problem using brute force.

As will become clear to the reader, the application of Systems Engineering tools to the TCAT routing problem proved to be very effective in:

1. Identifying the actual cause of low ridership levels
2. Clearly defining the user expectations from a new system
3. Identifying the characteristics of the system, tweaking which will result in an increase in the ability of the system to meet user needs
4. Clarifying the context and structure of the new system
5. Giving us a set of guidelines to help think of solutions
6. Giving us a framework to evaluate any suggested solutions based on their ability to meet user needs
7. Mitigating any possibilities of failure in the new system
8. Carrying out the whole design process in an effective and efficient manner

The report will essentially be a collection of tools and an explanation of their application to the TCAT Lansing Routing Problem.

Customer Value Proposition

Our first interaction with the client, in this case a representative of TCAT, did little to improve our understanding of the problem statement. It was a short meeting from which all we could gather was that the ridership levels are low and that is probably because the routes have not been revised in a long time. There were some hints as to the importance of new housing projects, the doctor's offices and the county jail to any new system that we came up with.

The customer value proposition is a document that helps you think about the problem from the perspective of the client. The key questions to be answered here are: Why should the client use your system and What about your solution makes it better than the competitors?

This guided us in thinking about the objective of TCAT in carrying out this whole project. It was to improve ridership levels without any capital expenditures or dramatic increase in operating costs. It was also our understanding that we needed to address their perceived reasons for the whole problem and to ensure the solution's relevance in the years to come. Any features that we would choose to include would have to meet these basic needs of the client.

This also led us to think about the question of who our client really was. Was it TCAT or was it the residents of Lansing? We concluded that it had to be both. The solution had to satisfy TCAT's needs for a low cost solution as well as meet the needs of the residents of Lansing.

This gave us a good starting to start asking better questions such as:

1. What would be a good solution from TCAT's perspective?
2. What are the transportation needs of the residents of Lansing?
3. How can these needs be fulfilled?
4. How do we know whether a proposed solution will actually fulfill these needs? In other words how do we test our solutions?
5. What are the various constraints that we need to work within while designing the system?

This led us to create the CVP that you can see in Appendix 1A. There are two main parts to the CVP:

- Log Line: This captures the essence of the value that our solution would create for the customer

- Body: This can be in paragraph form or as a list but it clearly maps the features we chose to include in the system to a user need. That is, it should be clear what feature has been included to satisfy which need.

Annotated Concept Sketch

The annotated concept sketch is an illustration of the system and its features in a visual form. It can help to highlight the important structural and functional features of a system. This is useful in conveying ideas about the system that would be difficult and time consuming to convey in words. It can help the audience get a sense of the salient features and how they relate to the needs of the user in a single glance.

For us this was more an exercise in starting to think about various ways in which the new system would be designed. It served as a warm up and got us thinking about the various ways in which the system could be made more usable for the residents of Lansing.

Another important outcome of this for us was that it got us thinking about what exactly was our “system”. Was it the routes and the schedules that the buses would take? Did it include the buses and the drivers? Would TCAT’s management team be a part of the system?

Although these questions were answered only when we got down to defining the context of the system in the context diagram section.

The annotated concept sketch is a pair of two sketches:

1. Structural Concept Sketch: This sketch focuses on the structural features i.e. the actual physical components of the system and not on how they relate to user needs.
2. Functional Concept Sketch: This sketch bridges the gap between the structural features and their role in fulfilling functional needs of the user

The concept sketches for the Lansing Routing System are included in Appendix 1B.

Customer Affinity Process

The customer affinity process helps identify major themes in user needs based on customer response data. An important prerequisite for using this process is the availability of user response. When we began the project we did not have access to any response data but towards the end we were able to gather significant responses and the team will work on the customer affinity process as the project moves into the second semester.

For the purpose of this report and to provide an example to the reader I have carried out this process on customer response data provided by Hasbro (attached in Appendix 2A).
The output of the Customer Affinity process is identification of the major themes and subthemes in user needs and their relative weightages based on their frequency of occurrence in the response data.

Context Diagram

The context diagram is a visual representation of the context that our system will have to work in. The context is anything outside the system that places any sort of constraint on it for example the weather, the local laws or even the wildlife.
The context diagram helped us answer the question that what exactly do we consider to be a part of the system. We decide that anything that we can directly control in the design process would be a part of the system. This led to the definition of the system as being a combination of the buses, the routes, the schedules, the bus shelters, the designated stops and the TCAT management team.
Anything other than this is the context of the system. The diagram helped us make sure that when we start thinking up solutions we do not end up with something that meets the user needs yet does not fit into the context. It can be compared with the constraints in an optimization problem. The solution to an optimization problem that is useful is the one that meets the objective while staying within the constraints.
The context diagram for this project is provided in Appendix 1C.

Use Cases

The use cases are essentially a list of ways in which the user might interact with the system. This may include interactions aimed at fulfilling a legitimate need or just at misusing the system. In our project this helped us make sure that we understood all possible interactions between the system and the users to make sure that the design we ultimately came up with would be able to fulfill all possible needs and also resist misuse of the system. In addition to this we also assign a priority to the use cases based on our understanding of how important a particular use case might be in ensuring a great overall user experience. This is useful in case we need to tradeoff between features that enable two separate use cases.

Thinking towards this particular objective in a targeted manner can reveal very interesting interactions with the system and help us design a system that is ready for any eventuality. This is a very powerful tool in making sure that the solution delivered to the client actually adds value and is robust. The list of use case is presented in Appendix 2B.

SysML Use Case Diagram

The list of use case referred to above is useful in itself but it can become very complicated to keep track of a list when it gets longer. Also it is difficult to visualize any patterns or connections that the use cases might have. For example it might happen that a use case is just an extension of another; or that one of the use cases includes another as a special case.

All these eventualities can be taken care of using the Use Case diagram. The diagram made it very easy for us to include a lot of special cases like a person travelling to the same location in the evening or in the morning. A person using the system in the evening might have very different needs than someone travelling in the morning. While this can be accomodated in a list, it becomes very long and unwieldy.

It is also useful when collaborating with people to be able to present these ‘use cases’ in a visual form as opposed to a list. For example our team consisted of designers and architects in addition to engineers. People with a lot of creative content in their work understand things better visually.

The use case diagram helped us convey the subtlety of differences in use cases to all team members very effectively. The SysML use case diagram is attached in Appendix 1D.

Use Case Behavioral Diagrams

The use case behavioral diagrams explore individual use cases in greater detail. Each use case is divided into logical steps that would be a part of the interaction represented by it. For each of these steps we add requirement(s) that the system would need to fulfill in order to make sure that the particular use case is successfully completed by the user. It is essentially a table divided into swimlanes. In our case the lanes are labelled “User” and “System”. The User column will list all the steps involved in a use case and the System column would contain a list of the requirements associated with each step. Below is an excerpt from one of the diagrams to help visualize this better:

Traveller uses the system to travel between office and home	
Initial Conditions	
1) System is ready for user	
Operator (Working Adult)	**System**
Gets to the position where he/she can start travelling to office using the system	
	Shall allow traveller to start travelling from a position that takes less than 15 minutes to get to from apartment_buildings_with_minimum_number_of_working_adults_to_justify_cost
	Shall allow traveller to start travelling within 30 minutes of leaving home
Gets off near office	
	Shall ensure that traveller reaches a position from where it takes a maximum of 15 minutes of walking to get to office 10 minutes before any office_hours

We used this tool to get a list of Originating Requirements which would help us design the best possible solution. If we design a system that adheres to the list of requirements drawn from behavioral diagrams for all possible use cases we are making sure that our system will perform well in any use case.

The use case behavioral diagrams for 4 use cases is included in Appendix 2C.

SysML Activity Diagram

The SysML activity diagram is similar to a use case behavioral diagram. It presents the steps that the user might need to perform to complete a particular use case while also showing what the steps that the system needs to perform in order to make sure that the use case is successfully completed by the user.

One distinct advantage of the SysML activity diagram is that it very clearly brings out the sequence of steps (to be performed by the user and the system). This allows the team designing the system to pinpoint triggers for initiating and deactivating a particular system feature.

We used the activity diagram to better understand the flow of use case steps for one particular use case.

It will be more useful going forward when features like GPS tracking or presence sensing might need triggers.

The diagram has been attached as Appendix 1E.

Originating and Derived Requirements Table

This tool essentially collates all the requirements into a single table, assigns an index to it and an abstract function name. The index makes it easy to reference them in documents and the function name makes it easy to discuss them.

The originating requirements are the ones that are derived directly from use cases. However, while making sure our system meets these we often encounter requirements that might be a prerequisite for meeting them. These are called derived requirements. They are included in a form similar to originating requirements except that it has an additional column containing the index for the originating requirement it was derived from.

This table is useful in keeping track of all requirements. It can be used to create a checklist for teams working on various subsystems so that they can keep track of all requirements.

The excerpts pasted below will make this clearer:

Index	Originating requirements	Abstract Function Name
OR.1	Shall allow traveller to start travelling from a position that takes less than 15 minutes to get to from apartment_buildings_with_minimum_number_of_working_adults_to_justify_cost	Office
OR.2	Shall allow traveller to start travelling within 30 minutes of leaving from starting position	Office, College

Derived Requirements			
Index	Derived Functional Requirement	Function Name	Source OR
DR.1	The system shall detect if there are users at a bus shelter within 3 minutes of a scheduled time for the bus	Shelter	OR.29
DR.2	The system shall send a late signal to the bus serving the route if there are users at a bus shelter within 3 minutes of a scheduled time for the bus	Reliability	OR.29

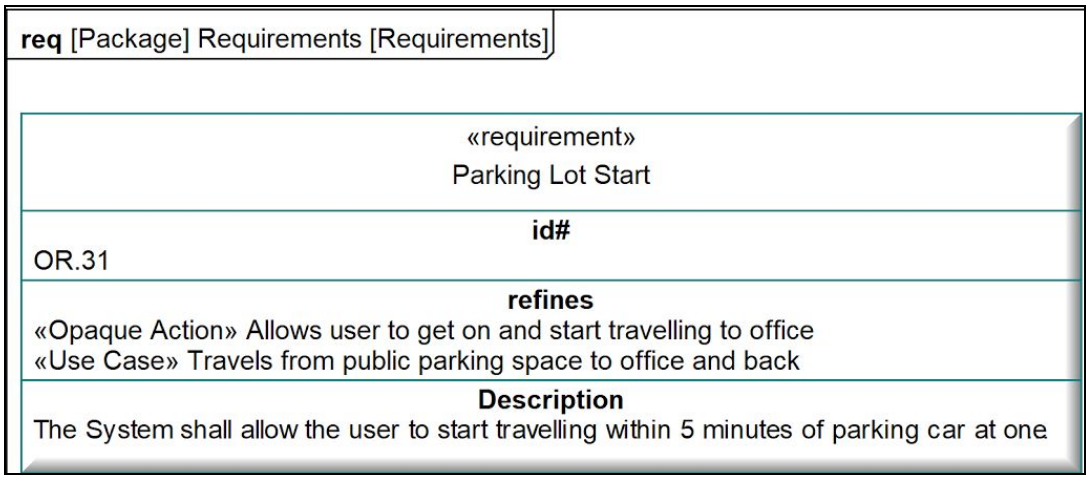
The full tables are attached as Appendix 2D.

SysML Requirements Table

The SysML requirements table enables us to associate the requirements to the steps involved in a use case in the SysML activity diagram. The SysML requirements table in conjunction with the Activity diagram is the equivalent of the use case behavioral diagrams.

This is a very powerful tool that automatically keeps track of requirements associated with a particular use case.

This table would contain an index number, the description, the abstract function name and the activity that the requirement is associated with. The following excerpt will help make this clearer:



The requirements table associated with the activity diagram attached as Appendix 1E is included with this report as Appendix 1F.

Concept Fragment Generation

This tool is meant to streamline the exercise of brainstorming for structural features that might be used to fulfill the originating and derived requirements. It takes the form of a list in which the requirement is copied at the top and the various ideas for fulfilling it are listed below. An example of this is included below:

Shall allow traveller to start travelling within 30 minutes of leaving from starting position
Distance between designated stops less than 7 miles
If more than 10 miles between stops on average add a bus
Shuttle bus or car from a fixed position to the designated stop

This would prove extremely useful in the next phase of system design when the team moves on to finalizing structural features of the system. The complete concept fragments are presented in Appendix 2E.

Combination Tables/Morphology Boxes

This tool can be thought as an extension of the Concept Fragment Generation. Although we may think up features that meet individual requirements we will ultimately need to pick the ones that are compatible with each other. Once we have brainstormed on the various ways in which we might fulfill a requirement we can use this tool to bring everything together by combining these concept fragments into a single table.

Moving into the next phase this will prove useful in narrowing down the list of features which fulfill the requirements. Below is a an excerpt showing two requirements and then their combined form and the combined features:

Requirement 1

Shall allow traveller to start travelling from a position that takes less than 15 minutes to get to from apartment_buildings_with_minimum_number_of_working_adults_to_justify_cost
Radial distance from all buildings less than 0.75 miles assuming walking speed of 3 miles an hour
If walking not possible need parking lot
Shuttle buses from apartments designated stop
Carshare
Bike Share
Driver with car to pickup people from all buildings farther than 1 mile
Add separate designated spot at a postion 0.75 miles from some buildings

Requirement 2

Shall allow traveller to start travelling within 30 minutes of leaving from starting position
Distance between designated stops less than 7 miles
If more than 10 miles between stops on average add a bus
Shuttle bus or car from a fixed position to the designated stop

Requirement 1 and 2

Reach designated stop within 15 minutes and start travelling in 30 minutes
Distance less than 0.75 miles and distance from previous stop less than 7 miles
Distance less than 0.75 miles but distance from previous stop more than 7 miles and more buses
Distance less than 0.75 miles but distance from previous stop more than 7 miles and shuttle bus to pickup point
Parking lot at designated stop and distance from previous stop less than 7 miles
Shuttle bus from apartment to designated stop and distance from previous stop less than 7 miles

The complete combination table is attached as Appendix 2E.

Functional Flow Block Diagram

The Functional Flow Block Diagram or the FFBD is a way to organize the use cases/requirements/steps from the use case behavioral diagram essentially anything functional related with the system. The FFBD clarifies the relationship between functions that the system must perform. As our understanding of the system grows and the list of use cases and requirements (essentially the functional features) keeps growing larger we need to be able to find relationships between them. We need to be able to define a hierarchy and a logic for the use of these functions.

An FFBD has various functions arranged in boxes and connected in a manner that signifies the flow of control between them. Very often FFBD's use logic gates to signify decisions of choosing to perform a specific function in order to effectively fulfill a functional need.

This is another tool that would greatly benefit the team in the next phase when it gets down to designing the system. It will help them trace the flow of the system's functions in parallel with the use case behavior. It can be thought of as tracing the steps presented in a use case behavioral diagram in terms of the system's requirements in the sense that when is the sub-system or structural feature associated with a requirement come into the picture.

The FFBD for the Lansing Routing System is attached as Appendix 1G.

IDEF0

This can be thought of as an extension of the FFBD. Whereas the focus in the FFBD is on the functional order the IDEF0 also makes the nature and content of interaction between the functions clear. The standard format of the IDEF0 has each function in a box and the functions are arranged in a hierarchical fashion. There are several arrows going into the boxes and coming out of them. The position of these arrows on the box signifies what they represent. The arrows entering from the left represent the inputs that the function would need and the arrow coming out from the right represents the output. The arrows entering from the top and bottom represent the Control/Constraint and the Resource/Mechanism, respectively.

This will prove very useful when defining the controls and the resources needed by the various functions. It will also help define when two functions interface with each other and what sort of information they exchange. This can help teams collaborate better if they know what input they

will be receiving and what is the output that is expected from each function for the next function to do its job. It will also be useful in preparing a Bill of Materials from the resource input to each of the functions.
The IDEF0 is attached in Appendix 1H.

Decision Matrices

Decision matrices are used to provide a quantitative way to make decisions about making a choice between alternatives. This is done by listing the set of criteria that are relevant to the choice being made. Then a weightage is assigned to each of them signifying their relative importance in the whole set. All of the choices are then scored based a pre-decided criteria. This can be the direct measure of a quantity in case it is something measurable. If it is not then a scale can be chosen (say 1 to 5) and a set of criteria are assigned to each of these ratings. These criteria must be fulfilled for the choice to be scored at a particular level.
The problem now is that all of these criteria will be on a different scale and we will need to normalize them to be able to combine them into a cumulative score. Once normalized these can be multiplied with the corresponding weightage and then added up to get a numeric score for each of the options. This can then be used to make a choice.
This is a very useful tool and will be used extensively to make decisions. It will be also be useful in quantitatively presenting the basis of any choice that is made throughout the process of system design. It gives authority to decisions and helps the team proceed with confidence knowing that their choices have a sound quantitative basis.
The various parts of a decision matrix are presented below:
The options and the set of criterion/attributes

	Value		
Attributes	*Current Route*	*Option A*	*Option B*
Coverage	100		
Total Stops	50		
Average Users per Stop	3		
Manpower/day	120		
Capital Cost	0		
Ease of Use	3		
Score Totals			

The attributes and their assigned weightages

Attributes	Weight
Coverage	2
Total Stops	4
Average Users per Stop	3
Manpower/day	3
Capital Cost	2
Ease of Use	1

The performance metric/attribute definitions

Performance Metric Definitions	
Coverage	The total area 200m on either side of routes in Lansing divided by the total area in Lansing. In square miles.
Average Users per Stop	The sum of people using a particular stop i.e. getting on and off.
Ease of Use	Combination of average distance to pickup point, average elevation change between pickup points and starting points, average number of bus changes required across routes that are used at least 5 days a week
Manpower/day	This is the number of hours that the drivers and the office staff have to work each day to ensure that the system is working well. Measured in manhours.
Capital Cost	This the cost of buying new buses or shelters or any other single or multiple expenditures above $50,000

The scoring criterion definition for a non measurable attribute

Ease of Use - Scoring Guideline	5	Average distance to pickup point <150m and average elevation change between pickup and starting point<10% and average number of bus changes=1
	4	Average distance to pickup point between 150 and 200m and average elevation change between pickup and starting point between 10 and 15% and average number of bus changes=2
	3	Average distance to pickup point between 200 and 250m and average elevation change between pickup and starting point between 15 and 20% and average number of bus changes=3
	2	Average distance to pickup point between 250 and 300m and average elevation change between pickup and starting point between 20 and 25% and average number of bus changes=4
	1	Average distance to pickup point >300m and average elevation change between pickup and starting point>25% and average number of bus changes>5

The full decision matrix is attached in Appendix 2F.

Goal–Question–Metric (GQM)

The GQM is something we found to be particularly useful in defining what exactly would be the goal of this new system that we would design for TCAT. It helped us define measurable goals and then the metrics that we could use to measure how well any solution would stack up against these goals. This is extremely useful as it provides a way to quantitatively evaluate any possible solution or feature in terms of the goals that the system has.

The GQM takes the form of table where the first column lists all the goals of the system. This list can be discussed with the client or compared with the RFP to make sure that if a system meets these goals it will most definitely meet all user needs.

The second column list questions that help us define these goals in a quantitative and measurable way by thinking about what it means to meet a particular goal. The third column

helps specify the ideal measure that would be used to evaluate a solution or answer the question(s) posed in the second column. It is not always possible to evaluate solutions on these ideal measures as the data might not be available or the method for collecting the needed data might be too time consuming or expensive so the fourth column presents list an approximate metric which would not be as good a measure as the ideal metric but it is something that is more practical to measure. And the last column is the proposed data collection method to be able to evaluate a solution on the approximate metric.
The GQM is attached in Appendix 2G.

Analytical Hierarchy Process

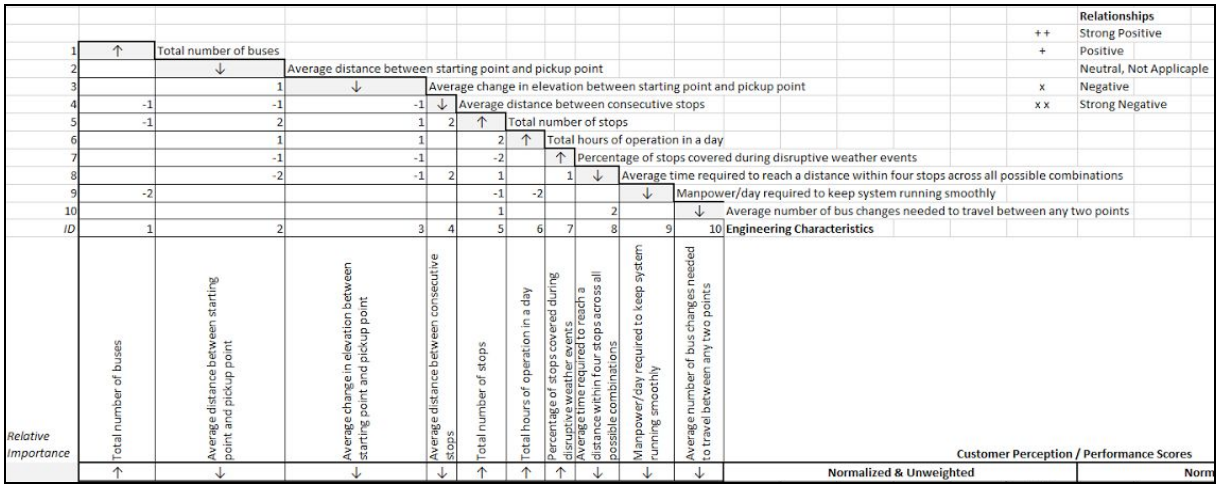
The GQM is essentially a way of streamlining and recording a brainstorming process but one key question it brings up is about the relative importance of the goals that have been defined. To think about this we used a tool known as the Analytical Hierarchy Process. This was used to identify the key themes in the goals like say reliability or ease of use. All the goals identified in the GQM are grouped according to these themes. Each theme was assigned a weight that signified its relative importance in the set of themes based on our own understanding and interactions with the client. These weightages must add up to one. This can easily be done by assigning a score to each of these on a pre-decided scale and then dividing by the sum across all the top level goals. Then the subgoals within each of these major goals are assigned a weightage that adds up to one within the major goal.
The analytical usually has 3-5 levels of hierarchy. The process of assigning weightages was continued till we had weights for all the goals we initially began with. The relative weightages of these goals were evaluated by multiplying the weightages at each level moving from top to bottom.
This helped us prioritize the goals of the system amongst themselves as to their importance in the system meeting the users functional needs. This will help the team allocate resources in a way that is proportional to the relative weightage of that goal.
The Analytical Hierarchy Process we completed for the Lansing Routing Problem is attached in Appendix 2H.

QFD (House of Quality)

Once we had all the product objectives from the customer's point of view and the requirements we needed something that would help us look how all these would fit together. There is always the possibility of an interaction between the engineering characteristics in the sense that modifying one might impact the others. We also needed to understand how much and in what way would the engineering characteristics affect the system's performance on the product objectives.

We needed to be able to modify one characteristic and immediately see how it impacted the systems' performance. For this we used a tool called the House of Quality or the Quality Function Deployment (QFD). The most important parts of a House of Quality are:

1. The engineering characteristics and their interactions: at the top of the QFD is a matrix of engineering characteristics which conveys the impact that modifying a particular characteristic has on the others. This is done by assigning an integer between -2 and 2 to each of the interactions. This signifies the strength and the direction of interaction. An excerpt from our QFD is presented below:



											Relationships
										++	Strong Positive
1	↑	Total number of buses								+	Positive
2		↓	Average distance between starting point and pickup point								Neutral, Not Applicable
3		1	↓	Average change in elevation between starting point and pickup point						x	Negative
4	-1	-1	-1	↓	Average distance between consecutive stops					x x	Strong Negative
5	-1	2	1	2	↑	Total number of stops					
6		1	1		2	↑	Total hours of operation in a day				
7		-1	-1		-2		↑	Percentage of stops covered during disruptive weather events			
8		-2	-1	2	1		1	↓	Average time required to reach a distance within four stops across all possible combinations		
9	-2				-1	-2			↓	Manpower/day required to keep system running smoothly	
10					1			2		↓	Average number of bus changes needed to travel between any two points
ID	1	2	3	4	5	6	7	8	9	10	**Engineering Characteristics**
Relative Importance	Total number of buses	Average distance between starting point and pickup point	Average change in elevation between starting point and pickup point	Average distance between consecutive stops	Total number of stops	Total hours of operation in a day	Percentage of stops covered during disruptive weather events	Average time required to reach a distance within four stops across all possible combinations	Manpower/day required to keep system running smoothly	Average number of bus changes needed to travel between any two points	**Customer Perception / Performance Scores**
	↑	↓	↓	↓	↑	↑	↑	↓	↓	↓	**Normalized & Unweighted** **Norm**

2. The performance objectives their relative importance and how they are impacted by changing the engineering characteristics. This is shown below:

ID	Performance Measures / Customer Attributes	Direction of Change	Relative importance	Total number of buses ↑	Average distance between starting point and pickup point ↓	Average change in elevation between starting point and pickup point ↓	Average distance between consecutive stops ↓	Total number of stops ↑	Total hours of operation in a day ↑	Percentage of stops covered during disruptive weather events ↑	Average time required to reach a distance within four stops across all possible combinations ↓	Manpower/day required to keep system running smoothly ↓	Average number of bus changes needed to travel between any two points ↓
ID	*Full Attribute/Goal/Add'l Info*	*Short Name*											
1	The total area 200m on either side of routes in Lansing divided by the total area in Lansing. In square miles.	Coverage	19.23	1	1	1	-1	2		-1	-1		
2	Performance metric representing the reliability of the route based on the number of buses running, distance between stops and the number of stops	Reliability	15.38	1	-1	-1	2	-1		1	1		
3	Average of total people using a particular stop i.e. getting on and off over all stops on the route	Average users per stop	11.54	-1	1	1			1	1	1		1
4	A performance criteria based on the distance from starting point to pickup point, change in elevation between starting point and pickup point, modes of payment available and the number of sources of route information	Ease of Use	19.23		2	2			1				2
5	Based on distance between starting point and pickup point, change in elevation from starting point to pickup point and number of buses	Senior friendly	7.69	1	2	2							
6	Based on distance between starting point and pickup point, change in elevation from starting point to pickup point and number of buses	Disabled friendly	3.85	1	2	2							
7	The total cost of using the system to travel for one month.	Cost of using system	15.38	-1	-1	-1	-1	-1	-2	-1		2	
8	Performance metric based on number of accidents in the past year, criminal incidents, fatalities and	Safety	7.69		2	1		1	-1	-1	1		1
	Competitor Legend	Measurement Units		Number	m	%	miles	Number	Hours	%	Minutes		Number

3. There are two other parts that the QFD has:
    a. The rating of our system and competitors on these characteristics to identify target values for each (alongwith an estimate of the importance of each, the technical difficulty and the estimated cost)

	Competitor Legend	Measurement Units	Number	m	%	miles	Number	Hours	%	Minutes	Manhours	Number
		Privately Owned Cars	NA	*5*	*0*	NA	NA	NA	NA	NA	NA	NA
		Carpools	NA	*5*	*0*	NA	NA	NA	NA	NA	NA	NA
		Private Shuttles	NA	*20*	*1*	NA	NA	NA	NA	NA	NA	NA
	B	Motorbikes	NA	*5*	*0*	NA	NA	NA	NA	NA	NA	NA
		(External) Requirement Thresholds	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
		Targets	**5**	**100**	**10**	**1**	**15**	**18**	**80**	**10**	**160**	**2**
		Imputed Importance	73	138	131	65	77	69	69	54	31	58
		Positive Imputed Importance	46	108	100	30.8	46	31	27	35	31	58
		Negative Imputed Importance	27	31	31	34.6	31	38	42	19	0	0
		Techincal Difficulty (1 Low, 5 High)	1	4	4	2	1	2	4	4	5	5
		Estimated Cost (1 Low, 5 High)	5	4	4	5	5	5	2	2	0	1

b. The comparison with competitors on the product objectives in the form of a decision matrix as shown below:

Customer Perception / Performance Scores									
Normalized & Unweighted					Normalized & Weighted				
TCAT Current Route	Privately owned Cars	Carpools	Private Shuttles	Motorbike	TCAT Current Route (Low)	Privately owned Cars	Carpools	Private Shuttles	Motorbike
3	5	5	5	5	57.7	96.2	96.2	96.2	96.2
4	5	5	5	5	61.5	76.9	76.9	76.9	76.9
1	NA	NA	4	NA	11.5	NA	NA	46.2	NA
2	5	4	3	5	38.5	96.2	76.9	57.7	96.2
2	NA	NA	NA	NA	15.4	NA	NA	NA	NA
2	NA	NA	NA	NA	7.7	NA	NA	NA	NA
3	2	3	3	2	46.2	30.8	46.2	46.2	30.8
4	2	3	3	1	30.8	15.4	23.1	23.1	7.7
					269.2	315.4	319.2	346.2	307.7

The QFD helped us think of target values of engineering characteristics that would help us best meet our product objectives. It also helped us think about how our system would compare with competitors and what features can or should be incorporated from them.
The full QFD is attached in Appendix 2I.

Sub-System Definition and Allocation

Once we had all the characteristics and objectives of the system defined we needed to get down to actually thinking about how we were going to work on the design. As with most teams we were a group of people with skills and expertise in very specific areas. We needed to identify parts of the system or subsystems to be able to split up the work into manageable parts and to split up as a team and work on parts that best suited our skills.
We used the subsystem definition and allocation tool to group functionalities and requirements into groups which would make sense to be fulfilled by the same subsystem. Once we had these subsystems we allocated the requirements to each of these subsystems to define what is expected out of each subsystem.
This will be useful on the next phase when the team will split up into teams and work on different parts of the system because they will have very clear outcomes in mind.
This tool is attached in Appendix 2J.

Operational Description Template (ODT)

Once we had the subsystems defined we needed to make sure that they can work together effectively. This involves an understanding of the role and interactions of each subsystem in while the system is completing a particular use case. This helps the subteams be aware of where and in what form can they expect their inputs and what outputs are expected from them for other subsystems to do their job effectively.

The operational description template helps divide use cases into functions that each subsystem must complete to ensure successful completion of a particular use case. Each column represents a description of the role of a subsystem in the use case and the general order in which control flows.

Whenever control flows from one subsystem to the other an interface row is added to indicate that there is some sort of interaction between subsystems at that point. Some text is added in the interface row which serves as a description or reference to a location with more detailed description.

This is the first step in making sure that the teams interface well with each other because being aware of the interfaces is the first step in managing them well. The ODT also has a column for the system state. This column is filled out for any step in which the system might change state. For example: at one point the bus is moving with its doors closed and when it stops at a bus shelter it will be stationary and have its doors open. This is an example of a state change.

The ODT for the Lansing Routing Problem is attached in Appendix 2K.

State Diagram and Matrix

Once we knew the states that the system will be in at different points in the process of completing a particular use case we needed a way to keep track of all these states and the triggers that initiate a state transition. This will help the subteams make sure that the system will be in the right state at for their subsystem to carry out its functions. For example: to initiate the bus door opening mechanism the bus needs to be stationary with doors closed. If it is moving or doors are already open this would present a problem or not make sense at all.

This was done by using the state transition matrix and diagram. The state matrix displays all possible state transitions and the trigger events for these.

The state diagram does this in a more visual form by having all states in separate boxes that are connected by trigger events.
The state diagram and matrix for the Lansing Routing subsystems are attached in Appendix 2L.

Interface Matrix

The operational description template defines all the interfaces between subsystems. This is very important for inter-subteam work so that there is no issue when integrating these subsystems because the subsystems would already have considered each other's requirements and constraints. The interface matrix is a way to arrange all the identified interfaces by different subsystems to make tracking easier. It has columns for defining the information exchanged at the interface. The direction of flow of information. The point of contact in the subteam responsible for the interface and like a version control column that helps track when was the latest change to the document made. It also has a reference to the document that contains greater detail on the interface and in case the information required is not yet finalized it contains an expected date and the person responsible for providing that information.
This is extremely useful when working in teams as it enables efficient collaboration by fixing responsibility to one person and keeping track of locations where information about the interfaces is stored.
The interface matrix for the Lansing Routing Problem is attached in Appendix 2L.

Behavioral Test Plan

Once we had a good handle on the various characteristics of our system, their target values, and the functional objectives we needed to be able to set very clear testing procedures that would be like checkpoints to verify if we are on the right track. A test plan that lists down all the details of testing procedures, the facilities needed, clearly defined initial and ending conditions and criterion for passing the test. It also maps the test plans to the requirements. What this means is that if a system passes that test we can be sure that the requirements associated with that procedure will be met. However this tool only makes sense for behavioral requirements (derived from behavioral details of the system during use cases and usually focus on things other than reliability and safety) which cannot be measured directly. For example, ease of use

cannot be measured but the number of buses running on a route does not need a behavioral test plan.

This is very important in any design process as it helps quantify and establish progress and helps us move ahead with confidence. Also we assigned unique index numbers to the test plans to easily refer to them.

The behavioral test plan for the Lansing Routing Problem is attached as Appendix 2M.

Test Methodologies for Non-Behavioral Requirements

Non-behavioral requirements (dealing with reliability and safety) that are essentially specifications of one form or the other are grouped in a separate table for testing methodologies for non-behavioral requirements. This table is almost similar to the Behavioral Test Plan with the exception of two additional columns mentioning the index of the requirement and a classification of the test procedure as one of the 4 categories:

1. Inspection (I): if the methodology mainly involves visual examination of the system, measurement and comparison with some standard to ensure conformance
2. Analysis (A): if some established and widely accepted procedures, algorithms, simulations or models to test the system
3. Demonstration (D): if the test involves the actual operation of the system. Used when no quantitative assurance is required.
4. Test (T): if the testing requires designed procedures to collect data meant for verifying the fulfilment of the requirement.

The behavioral and non-behavioral testing plans are critical to building a system that meets user needs within the budget and time constraints. If designed and followed properly these will make sure that we do not end up with a system that fails to meet user needs and requires a large amount of rework. While building these test plans we were also able to confirm that all our requirements were verifiable and we can actually design tests to be sure that they are being met. If this had not been the case we would have tried to refine the requirement or break it up into multiple requirements to make sure they were verifiable.

The test methodologies for the non-behavioral requirements are attached in Appendix 2N.

Verification Cross Reference Matrix (VCRM)

The two tables above gave a long list of testing procedures that would ensure that our system adds maximum value for the client. However, it is difficult to keep track of all these test plans in the form of a table. For this reason we turned towards the VCRM. This is a matrix that traces each test procedure referred to by its unique index to the originating requirement, functional requirement or the derived requirement that it helps verify.

Besides keeping track of all procedures this also helped us be sure that we had test plans in place for all the requirements. Because if we miss even one requirement we can never be sure whether our system meets that requirement or any derived requirements.

The VCRM for the Lansing Routing Problem is attached in Appendix 2O.

Severity Rating System and Likelihood Rating System

The severity rating system is a tool that is used for risk classification when carrying out a risk minimization known as the Failure Mode and Effects Analysis (FMEA)—discussed below. Despite all our efforts there is always the risk of something going wrong with the system. So we as a team decided to put efforts into recognizing any possible ways in which the system could fail and putting measures in place to either prevent them or mitigate their effects. For this reason the effects of any sort of failure need to be prioritized to allow us to focus our limited time and resources on the most damaging ones. This was done by assigning a severity rating and a likelihood rating to each of the failure effects and failure modes, respectively.

To do this we first need to decide a scale for rating the severity and the likelihood of occurrence and then have very clear guidelines/criterion as to the assignment of these ratings. The severity represents the amount of negative impact expected from a failure effect and the likelihood of the failure mode i.e. the odds of occurrence.

If any past data is available the likelihood can be estimated using a reference distributions that fits the data well (by using the Mean Time to Failure(MTTF) or the Mean Time Between Failures (MTBF)) and the severity can be judged from past occurrences. We did not have access to data so we have used the team members' experience and judgement to define these rating system. The systems are presented below and are also attached in Appendix 2P.

Table 2 Severity Rating System

Severity	
5	Causes a loss of more than 30% in ridership or a capital loss of more than $10,000 or both
4	Causes a loss in ridership between 20 and 30% or a capital loss of $8,000—10,000 or both
3	Causes a loss in ridership between 10 and 20% or a capital loss of $6,000—8,000 or both
2	Causes a loss in ridership between 5 and 10% or a capital loss of $4,000—6,000 or both
1	Causes a loss in ridership less than 5% and a capital loss of less than $4,000

Table 3 Likelihood Rating System

Likelihood	
5	Occurs more than once a week
4	Occurs more than once a month
3	Occurs once a month
2	Occurs once a year
1	Occurs once in 10 years

Risk Priority Number Table for Criticality Rating System and Stoplight Graph

Once we have the likelihood of the failure mode and the severity of its effects we need to be able to combine them to give us a single number quantifying the risk of failure associated with each failure effect. This product is called a Risk Priority Number (or RPN). It is presented in the form of a matrix with severity and likelihood on the axes. This not only gave us a way to prioritize the effects it also refined our thinking process about the design features that we would include in our system design to enable it perform the functions needed of it.

A cut-off RPN can be decided which would mean that the team will focus all its efforts in bringing the failure effects with an RPN below this threshold for the system to be considered reliable.

The range of RPNs can be divided into several parts and labelled using ordinal categories and is assigned a color as shown below. The matrix of RPNs is then colored according to these.

The full FMEA alongwith the RPN table and the Stoplight Graph is attached as Appendix 2Q.

Figure 8 RPN Table and Stoplight Graph

Risk Priority Number (RPN) Definition Table							Risk Criticality Ranges	
Likelihood								
5	5	10	15	20	25		16-25	High Risk
4	4	8	12	16	20		10-15	Medium High Risk
3	3	6	9	12	15		5-9	Medium Risk
2	2	4	6	8	10		3-4	Medium Low Risk
1	1	2	3	4	5		1-2	Low Risk
	1	2	3	4	5	*Severity*		

Failure Mode and Effect Analysis (FMEA)

The FMEA is a very widely used tool for risk assessment and minimization. It takes the form of a table which contains:

1. possible failure modes
2. effects of those failure modes
3. Likelihood of the failure mode
4. Severity of failure effects
5. RPN
6. Possible causes
7. Proposed corrective and preventive actions
8. Priority base on stoplight graph and decided threshold of RPN
9. Estimated or actual RPN after implementation of suggested measures
10. An additional column is sometimes present which affixes responsibility for a particular failure mode to an individual(s). This individual is usually different from the one who pointed out the mode in the first place.

The FMEA proved to be very critical in helping our designing process to take into account these risks at the very beginning so that the first pass design itself would be very reliable. Especially in phase 2 of this project the FMEA will prove extremely useful by ensuring that we design for reliability.

The FMEA for the Lansing Routing System is attached in Appendix 2R.

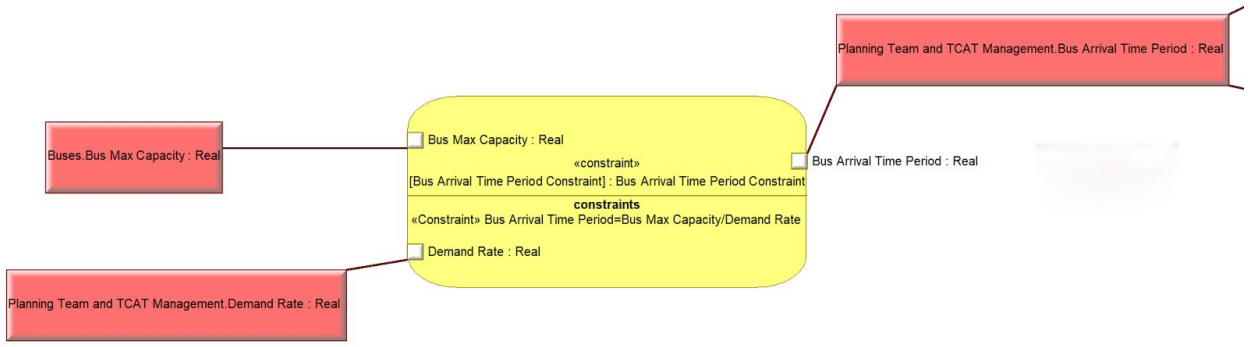
Event Tree or Fault Tree

The fault tree is another tool that we used to identify the risks associated with our system. It is a top down approach to risk identification. How it work is that we start with a top undesirable event that poses a risk to the system's performance or to the user. We then move downwards branching out the top events into sub events that may lead to the occurence of the top event either individually or in conjunction with each other i.e there is either an OR gate connecting them which means that the occurrence of either one of these sub-events would lead to the occurence of the top event or an AND gate which means that for the top event to occur all the subevents need to occur at the same time. Then we keep dividing these sub-events further till

we cannot possibly divide it further i.e. we have reached the equivalent of first principles for the system. The events at the base are called root events. Then we assign probabilities of occurrence to each of these and keep combining them using the rules of probability addition for AND and OR events to get to the probability of occurrence for the top event.
This not only helped us get an idea as to the likelihood of the occurrence of a top level undesirable event but also the possible causes so that we could be extra careful while working on those parts of the system.
The fault tree for the one particular top level event that we thought important is attached as Appendix 2S.

SysML Parametric Diagram

Each of the subsystems we defined takes some input and produces an output that might serve as the input to a different subsystem or be the output of the system (as made clear in the IDEF0). In either case these inputs and outputs usually obey certain mathematical rules/equations due to the design of the system. Any system is designed to take some inputs perform some operations and give us an output. The SysML parametric diagram helps formalize these constraints into a model and presents the inputs, outputs and the constraint itself in a very visual form. In the sense of the SysML model itself it would help us during simulations of the dynamic behavior of the system by letting the Modelling Software know how inputs relate to each other or the outputs.
Besides getting the model ready for computational use it helped us in thinking about building the constraint blocks helped us better understand the relationship between various system blocks. This improves the designing process because we knew how an output relates to its inputs and the magnitude and nature of modifying an input might have on the output or the input itself so that we could make better decisions.
An excerpt from the full parametric diagram is presented below and it shows the inputs the outputs and how they relate to each other.



The full parametric diagram having 3 constraint blocks is attached in Appendix 1I.

Timeline Update or Extension

The project timeline is probably one of the most commonly used tools for managing projects. We used it to divide the project into small steps/milestones and set timelines for them as well as responsibilities for individual steps. The most important part of a timeline however is making sure that one step is completed before the team moves onto the next. This is done by means of clearly defined deliverables that must be completed and also a procedure for the deliverable's verification to be sure that it is in accordance with client requirements in user needs.

The timeline for the Lansing Routing System Design is attached in Appendix 2T.

We had divided the project into phases with the first phase being understanding the problem and defining the system that we need to design. As next steps I would recommend getting divided into teams based on the subsystems outlined in the subsystems and start working on designs that will fulfill the requirements allocated to each subsystem.

The work can be streamlined using the interface matrix to make sure that the integration happens smoothly and the test plans and methodologies need to be followed to ensure the progress of the design in the right direction.