Modeling the CHIPSat System within the Alpha Project

Philip Naumann
Net ID: pn246
Systems Engineering, M.S. ‘22
Engineering research in collaboration with Josh Umansky-Castro

Special thanks to Dr. Mason Peck, Dr. David Schneider, and Mr. Wes Hewatt

Table of contents

Modes used for Analysis of the CHIPSat within the Alpha Project

How would you attempt to send physical satellites at over a quarter of the speed of light towards other solar systems in search of life-sustaining planets? The Alpha project, as part of Cornell Space Systems, aims to solve this challenge using a complex system containing a rocket, Cubesat, and several CHIPSats. However to guarantee success, a system such as this requires extensive systems modeling, outlined by the International Council of Systems Engineering (or INCOSE). The following paper will examine each of these models, explain how they are used, discuss the model's defining characteristics, analyze the implications revealed about our system, and elaborate on any further design that was prompted. This paper reviews the systems engineering models listed in the table of content chronologically and discusses and demonstrates their use as applied to the Alpha project for the modeling of the CHIPSat embedded microsatellite. Also, all of these models referenced can be found in full. The table of contents navigates to the corresponding page number of each model in the appendix.

To summarize the nature of the Alpha project, the mission was prompted as a proof of concept of the light sail theorized in the Breakthrough Starshot project. Using this light sail, it was theorized that an embedded microsatellite on the order of a few grams could be accelerated to a quarter of the speed of light in a matter of seconds. If achieved, this technology would be revolutionary, because it would broaden exploration into horizons beyond our solar system. This would open doors to research on a whole new magnitude. The project that this paper will refer to, however, is the specific design of the CHIPSat embedded microsatellite (that is being propelled by the light sail) on both the hardware and software fronts.

Customer Value Proposition

The first model as outlined by the table of contents, is the Customer Value Proposition (or CVP). The CVP is designed to give a snapshot about the project in order to attract investors. To be able to do so effectively, the CVP is designed to be short and concise. The CVP is intended less as a pitch, and more as a synopsis of the project. The CVP should go through the plan for the project, giving the important details and the

value that the system brings to its target audience. Using the CVP, helped condense and summarize the project into a manageable “elevator conversation” length description. This way the key components of the mission were highlighted, giving a broad context to the project at hand. This prompted analysis into the feasibility and lifecycle of this system, which in turn fueled the rest of the models in this paper.

Annotated Concept Sketch

One of the very first tools used in modeling the CHIPSat was the Annotated Concept Sketch. As is intuitive by it’s title, the Annotated Concept Sketch is a concept sketch that is annotated to show design details and functionalities. In short, it is a mockup of the initial design, with notes to clarify all of the parts and pieces. These notes are split up between structural components and product functions to avoid confusion. The Annotated Concept Sketch is useful as a first representation of the concept stagaty. Albert Einstein once explained “If you can't explain it simply, you don't understand it well enough”. Reflection on this quote, the Annotated Concept Sketch serves as an explanation; if you aren't able to make a simple concept sketch, maybe you don't fully understand the system yet. For the CHIPSat system, making an annotated concept sketch proved a basic understanding of the elements that comprise the CHIPSat. Making this sketch prompted a need for the Customer Affinity Process and the Requirements Diagram to specify the specific parameters of the design.

Customer Affinity Process/ Stakeholder Request Table

The next model used was the Customer Affinity Process. Due to the research nature of this project, this model can be exchanged with the Stakeholder Request Table. This table lists any requests made by stakeholders to guide the development of the project in the direction that would be in the company's best interest. After using this model, the project scope can be defined by discussing the possible use cases and markets for the product. Once a customer and market has been solidified, the stakeholder requests are turned into requirements of the project. In essence the requirements act as the contract of the mission. For the CHIPSat, the Stakeholder

Request Table clarified expectations for the project. This request table functioned as a less visual version of the Context Diagram.

Context Diagram

After having established the stakeholders vision for the project, a Context Diagram can be made to to clearly define the scope of the system, and get a perception about other “neighboring” systems. The Context Diagram is a block diagram made by placing the system of interest in the center of the diagram in a square box (WITH SHARP CORNERS), and adding horizontal or vertical (or a combination thereof) lines that connect to other systems. A dotted line is placed around the edge of the system. This defines what will be considered in the system and what is considered external to the system. Also the Context Diagram clarifies what other systems this product will interface with. For example, in the CHIPSat design, the users, sponsors, associated organizations or other physical systems were included. Because the CHIPSat system directly interfaces with radio transceivers, the CubeSat, and NASA’s regulations, each of these interactions will mold a certain design interface. For the CHIPSat design, the radio frequency had to be set in compliance with a bandwidth given by NASA. In addition, the shape and size of the satellites had to be made to fit seamlessly into the rocket. Using the Context Diagram, a clear picture is painted of what kinds of constraints the inputs/outputs of the system will be held to. The Context diagram is also a good way to brainstorm the Use Cases of the system of interest.

Use Cases

Moving on, the Use Case Diagram is used to come up with all of the possible uses of the device and the corresponding priority level. This not only applies to the functions of interest within the system, but also any to the use or abuse of the product that may occur in the environment specified by the Context Diagram. For example, if the device is hand held, a use case could be dropping the device. Brainstorming all of these Use Cases, helps a designer recognise both the priority of each of its functions, and the importance of creating preventative measures such that the product does not incur

damage if abused. For the CHIPSat, the Use Case Diagram encouraged the restructuring of tasks based on how important they would be to the mission. In addition it shed light on risks associated with the environment like radiation, vacuum, low temperatures, and space debris. Because of early identification of these risks, a solution could be incorporated in the design process. The Use Case Diagram Is a broad diagram that is relied upon by many other systems engineering models. Some extensions of the Use Case Diagram include the Use Case Behavioral Diagram and the Use Case Activity Diagram.

Use Case Behavioral Diagrams

The Use Case Behavioral diagram is very similar to the Use Case Diagram, however it focuses on the system interaction with its user/s. In this diagram, relations are drawn between the user and the appropriate use case. Each use case is also given a priority level based on a colored legend. This diagram differs from the original Use Case Diagram, because it is able to show separate use cases for different users of the system. Also, because of the diagram nature of this model, the interactions can be further specified visually. For example, a dashed line pointing to the left can show how a use case extends, while pointing to the right shows inclusion. For the Alpha project, using this diagram helped specify how the researchers may interact with the product differently than how the operators would. The operator could be assigned use cases pertaining to use and data collection of the CHIPSat, while the researchers use cases could include packaging or communicating with the CHIPSat. This allows for a different understanding of the use cases in separate contexts of its use. The use case diagram outlines some of the relations that are elaborated on in the activity diagram.

Activity Diagram

The Activity Diagram considers a specific use case, and walks though all the progression that must occur for that task to be completed from beginning to end. Coulombs are made for every subsystem involved in the process, and subtasks fill the rows such that no more than one subtask exists in any row. This can be seen in Figure 1 below.

Opperator (Researcher)	System (CHIPSAT)	Accelerometer x3
Researcher sends command on computer for CHIPSAT to record the acceleration		
	Radio wave received by CHIPSAT	
	Command processed by CHIPSAT Software	
	CHIPSAT pins gets ready to read data from the gyroscope	
		Accelerometer returns the acceleration of the CHIPSAT along each axis in 3D space
	CHIPSAT receives digital data in the form of a vector with a dimension of 3	
	CHIPSAT stores data and sends a copy over radio waves	
Acceleration data is received as an electromagnetic wave and converted to digital data the researcher can interpret		

Figure 1: Activity diagram showing energies in the rows and columns that specify the progression of tasks

Looking at Figure 1, it is apparent that the functions are recorded in the matrix such that the subsystem is identified by the colomb, and the order is apparent by the row. This makes it easy to track the progression of a task by starting at the top of the diagram and weaving through the coulombs, down the table. For the CHIPSat, defining what subtasks each subsystem was responsible for, made it simpler to organize how the software was written. The Activity diagram also serves as the basis of the Operations Description Template (ODT), however the states have yet to be defined.

Originating & Derived Requirement Tables

Next, Requirements Tables can be made from the Customer Request Table. The Requirements are split into two tables, the originating requirements, and the derived requirements. The originating requirements are discussed and agreed upon between the shareholders and the engineers. These requirements become the criteria for which the product will be judged. Therefore it is important that all of the requirements are measurable and have quantitative tangible deliverables. The requirements in the Original Requirements Table become the contract-of-sorts to determine if the product is valid. Failing in meeting the originating requirements would be a breach of contract, or failure to create the agreed upon product. If a requirement is not able to be met, a

compromise must be reached between the stakeholders and engineers. The originating requirements make it clear what is expected in order for the product to be a success. After the originating requirements are determined the next step is setting the derived requirements.

The derived requirements are set as a result of the originating requirements. The derived requirements are made by the engineers such that engineering goals can be made for the system. These derived requirements are made by “reading between the lines” of the originating requirements, and determining the appropriate engineering standards needed in order for the system to work as agreed upon. In short, derived requirements are the goals that must be met such that the originating requirements can occur. This could mean determining tensile or shear strengths of certain components, evaluating the appropriate current and voltage in a circuit, or requiring a certain processing power in a microchip.

In reflection, the originating requirements are generally broad requirements that outline how the system should properly work. The derived requirements are more difficult to set because they constrain the design such that it is guaranteed to meet the originating requirements. This can create difficulties because it could force the redesign of certain parts. In addition, coming up with these requirements prompted thoughts on how a system would be tested. In the Alpha project, it was difficult to simulate outer space while testing. This meant splitting up and testing the environmental threats individually. In contrast, the CHIPSat software was very testable. This made meeting the requirements very straightforward. Sometimes an originating requirement could prompt two or more different design ideas, in which case there would be multiple solutions.These solutions can be written as Concept Fragments, and arranged in a Combination Table.

Concept Fragment Generation/Combination Tables

For a system with several parts and subsystems, there are always multiple methods to completing each task. Concept Fragments are derived by brainstorming the different design ideas and listing them in a table. Ideas that conflict with each other go in the same column. This way the different methods of achieving the required tasks can be

seen in each column. Next, a Combination Table is made with all of the possible combinations of the Concept Fragments. Making a Combination Table is insightful in discovering new ways that different approaches could be used in coherence. It is important to keep in mind that some of these ideas still may not work, or may be unnecessarily challenging. The point is to find feasible solutions that could interface well together. Using Combination Tables for the CHIPSat design was insightful, because it offered a big picture view of some of the possible design solutions. Creating a Combination Table became overwhelming with combinations of more than two or three different variables at a time. Combination tables sparked the need to define the flows between each of the components. This was elaborated on in the Functional Flow Block Diagram (or FFBD).

Functional Flow Block Diagram

The Functional Flow Block DIagram (FFBD) is one of the hallmarks of system engineering modeling, and is used for System Architecture. This diagram starts off with a reference, and follows the flow of a resource, such as time, through all of the corresponding system functions until it reaches a final reference point. As the name suggests, each of the functions are depicted as blocks and connected in a progressive flow. Each block is numbered chronologically and given a function reference number. This can be seen in Figure 2 below.

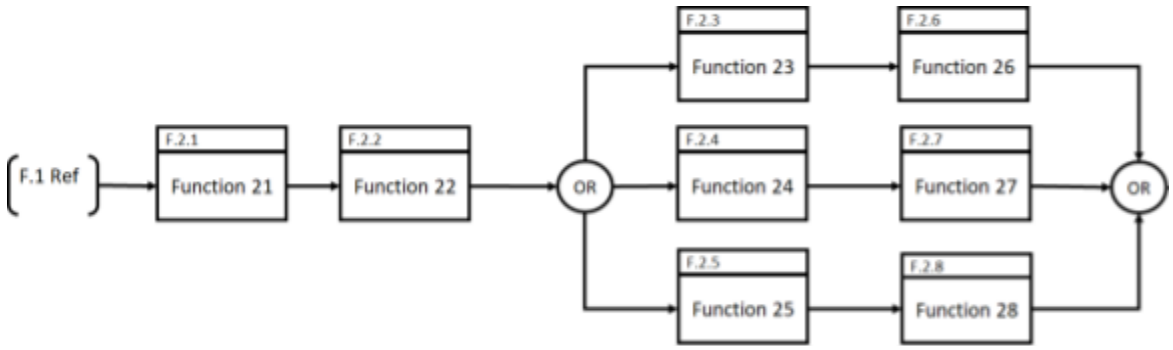


Figure 2: FFBD example. Functions are depicted in chronological order with the reference number labed in the upper left hand corner

The reference number found in the figure can be traced back to the FFBD tabel. This table contains the actions associated with each of the reference numbers. This

diagram made clear the progression of functions that take place in the system. Using an “or” divider can show decision making capabilities of the system and likewise, using an “and” divider indicates two flows working simultaneously.

Incorporating the FFBD in the CHIPSat modeling effort added a defined flow of operations. The FFBD provided an easy-to-read block diagram where the logic flow was clear. This was critical in modeling the functions of the embedded CHIPSat, because it built a foundation for the organizational structure of the code. For example, being able to model the synchronist and alternating structure of the sensors shed light on how a Real Time Operating System (RTOS) could be used to queue different sensors at a time. This would make it possible to collect real time data from many different sensors which all carried a similar timestep. Following this model, a different flow diagram like the ICAM DEFinition for functional modeling (IDEF0) could be implemented. This would be able to elaborate and distinguish between different flow types.

ICAM DEFinition for functional modeling (IDEF0)

The IDEF0 is another Systems Architecture flow diagram that is widely used in systems engineering. It has many similar implications to the FFBD, however it focuses more on the flows than the functions. For example the IDEF0 categorises the flows such that the inputs enter the system to the left of the blocks, with the outputs exiting to the right. In addition, any controls or constraints are drawn entering the system from the top. If the system uses a resource, but the resource serves as a tool and is not consumed, it can be drawn coming into the bottom of the box. Another strength of the IDEF0 is that it is split up into different magnification levels. This way, one block diagram can exist for the overall system. More complicated subsystems can be split up into their own models on a lower level of the diagram. This allows for the modeler to dig deeper into certain regions of higher complexity without wasting time having to recreate the model on every level. It is common for an IDEF0 model of a complex system to even have three or four different levels demonstrating more intricate details of the design.

Below, in Figure 3, is the top level of the IDEF0 used in the CHIPSat design. In this diagram, all of the inputs of the system are clearly shown.

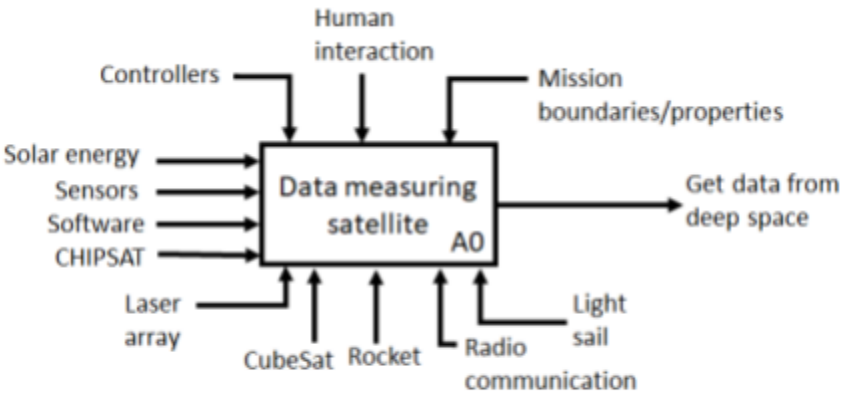


Figure 3: The top level block diagram of the CHIPSat IDEF0. Here all the inputs and outputs of the system can be seen

Following this IDEF0 to the next level, it is clear that for the overall system, the overall net inputs and outputs must stay the same. This can be seen in the second level of the IDEF0 depicted in Figure 4 below.

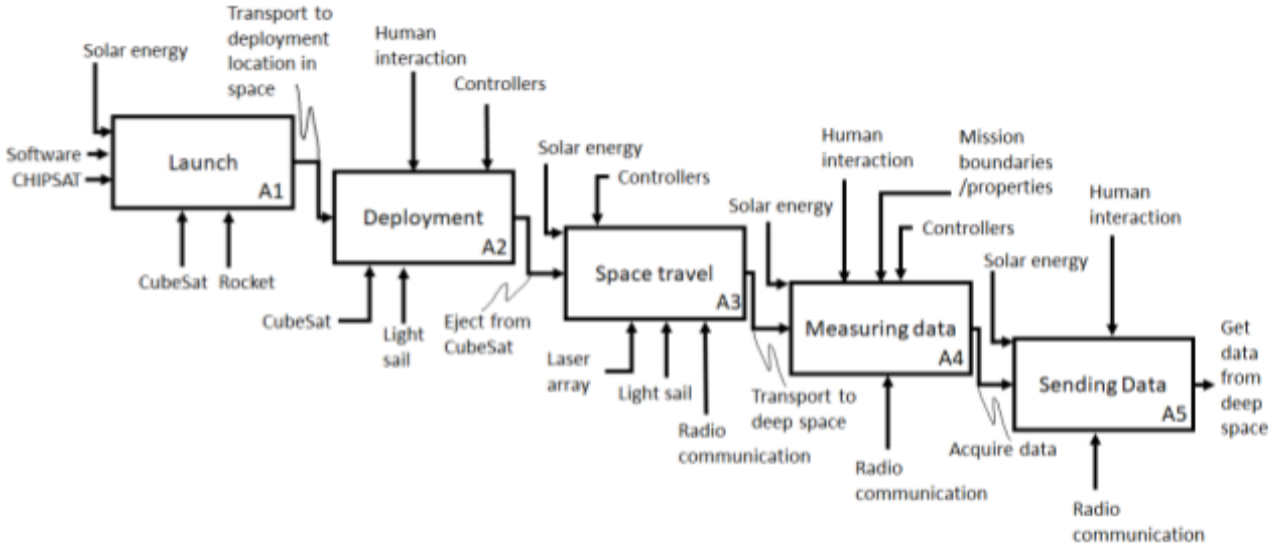


Figure 4: The second level block diagram of the CHIPSat IDEF0. Note the overall inputs and outputs of the system stay the same

The IDEF0 is a great model because of its versatility. It is able to show a broad overarching scope of the entire system, while capturing the trivial details of complicated subsystem assemblies. For the modeling of the CHIPSat, the IDEF0 helped define the flow of different resources in and out of it. In addition, by using the IDEF0 model, each

resource was tracked such that in steady state the inputs and outputs were balanced. In the CHIPSat, this technique verified that each interaction was modeled correctly.

In addition, using the IDEF0 also clearly defined when and where the system would use a resource. For example, looking back at Figure 4, it is clear that the lasers are not needed until the system reaches the space travel phase. It is also clear that this phase is after the satellite deployment but before the measuring data phase. For the CHIPSat software, this information was crucial. Being able to structure the code by both the time interval and system was extremely beneficial.

Decision Matrices

Changing course, many times in engineering it can be important to make decisions objectively by weighing all of the options. One way systems engineers are able to model the decision making process is though the Decision Matrix. The Decision Matrix is used to weigh different options based on their tangible qualities. These tangible qualities must first be defined in a Decision Scale Definition Matrix. This matrix is used to scale all of the qualities of interest on a range from one to five. This allows the options listed in each column of the Decision Matrix to be evaluated as defined in the Decision Scale Definition Matrix. Each of the criteria are then given a weight based on the believed user dependency and rescaled. A minimum score for each of the qualities is made such that options that do not meet the criteria are given a score of zero. The final score of each option is then computed by summing the adjusted scores earned. The highest score determines the option of choice.

This model was used in determining the best CHIPSat prototype to pursue in development. As can be seen in Figure 5, the different models were labeled A, B, and C. The criterias of interest included the resolution of certain sensors, the update frequency, lifespan, and cost. Based on this matrix a decision was made to go with option A.

	Normalized score			User Dependencies		Final Score		
	A	B	C	Min Score	Weight	A	B	C
Temperature resolution Sig figs (1-5)	3	2	5	3	4	12	0	15
Pressure resolution Sig figs (1-5)	4	3	3	3	4	16	12	12
Light sensor resolution Sig figs (1-5)	4	2	3	2	3	12	6	9
Update info frequency 1-every 5s -> 5-every 1s	2	5	4	2	1	2	5	4
Average lifespan 1-10years -> 5-50years	3	5	4	2	5	15	25	20
Cost 1-9k,2-8k,3-7k,4-6k,5-5k	5	4	2	1	2	10	8	2
						67	56	62

Figure 5: The Decision Matrix used in choosing a CHIPSat Design. This matrix verified that option A was most desirable

The decision matrix was an extremely useful tool in designing the CHIPSat. Many strategies used in the decision matrix were applied in multiple other facets of decision making during this project. One example of this would be the cost benefit analysis, among other things. Naturally, the decision matrix foreshadows the much more complex Quality Function Deployment (QFD) that will be discussed later.

Goal Question Metric (GQM)

Another important model that is used in evaluating system performance is the Goal Question Matrix (GQM). This matrix is used to develop testing strategies with which systems can be measured and evaluated. The first step in creating a GQM is developing goals based on the requirements that need to be tested. These goals are made during the development phase to ensure that the design of the product is robust and reliable. Testing early and often is a crucial part of the development phase because, as the product moves through it's lifecycle processes, the cost for corrective action increases substantially. Once goals are determined, appropriate testing strategies can be developed. This is done by noting any relevant questions that would reveal the status of the goal. For each question, a corresponding ideal metric, approximate metric, and data collection method is attained. Using the finished GQM, a

tester should be able to follow the data collection method and calculate a metric that would evaluate the status of the goal.

For the CHIPSat model, the GQM assisted in creating a testing plan. This split up the testing and evaluation process of each of the individual elements. Using the GQM helped define and pinpoint problems with the CHIPSat software. For example, when testing the CHIPSat transceiver, it was important to single out the receiver and transmitter signals individually. Breaking the requirements into attainable metrics made finding issues more manageable. Creating the GQM gave rise to future testing techniques used in the building of a Behavioral Test Plan and Test Methodologies of the Verification Cross Reference Matrix (VCRM).

Analytical Hierarchy Process

When designing a product, allocating time and resources effectively can be difficult. Using the systems engineering Analytical Hierarchy Process is useful in determining the respective weights of different components of the project. The analytical hierarchy process is made by listing the major subsystems in columns of a matrix. Normalized weights are assigned to each one. Going down the rows, each subsystem is further dissected into it’s respective components, each of which are weighted again. Finally, the relative importance is computed by multiplying down the column. Using this metric, each subpart can be objectively ranked (see Figure 6).

	Make the satallite able to communicate over radio		Make the satalite able to power itself	Make the satalilight able to collect data				Make sure user can identify if data is accureate
1	0.25		0.35	0.3				0.1
2				Kinematics of satalite		Extereior measurements		
				0.45		0.55		
3	Make the satallite able to recive commands (RX)	Make the satallite able to send data packets (TX)	Make the satallite able to power itself	Make the satallite able to measure acceleration along all 3 axies	Make the satallite able to measure position in 3D space	Make the satallite able to measure ablsolute pressure	Make the satallite able to measure ablsolute temperaure	Make sure user can identify if data is accureate
4	0.2	0.8						
5			1					
6				0.25	0.75			
7						0.5	0.5	
8								1
9								
10								
11								
12	0.05	0.2	0.35	0.03375	0.10125	0.0825	0.0825	0.1

Figure 6: The Analytical Hierarchy Process depicts the relative importance of each of the components as calculated by multiplying down the coulombs

The Analytical Hierarchy Process is a simple way to determine the relative importance of each of the components in the system. It makes the priority of each of the functionalities very clear, resulting in better time management and budgeting of resources. The relative importance calculated in the Analytical Hierarchy Process is also insightful for rating severity in the risk analysis.

Quality Function Deployment QFD (House of Quality)

Possibly the most complex diagram used in Model Based Systems Engineering is the Quality Function Deployment (QFD). The QFD is used to measure the quality of your product in comparison to other market options, as perceived both quantitatively and by the customer. There are six parts to the QFD; the Customer Objectives, Customer Perceptions, Engineering Characteristics, the Impact of the Engineering Characteristics on Customer Objectives, Interrelationships, and the Targets. Beginning

with the Customer Objectives submatrix, the customer focused product objectives are listed. These are found from the Customer Affinity Process. The relative importance of these can be inputted by referencing the Analytical Hierarchy Process.

The next step of the QFD is completing Customer Perceptions These are evaluated using a Decision matrix. The corresponding Decision Scale Definition Matrix required is derived by creating a GQM. This evalues the Customers Perceptions of the product of interest and its competitors as judged by the Customer Objectives derived in the first part. Using the Decision Matrix, the product is judged on how competitive it will be in the marketplace.

In the next step of the QFD, Engineering Characteristics that have the potential to affect the Customer Objectives are listed. The Engineering Characteristics can be measured and compared directly to competitors in the Targets section. Typically Engineering Characteristics are evaluated by cost, difficulty, strength, or other engineering abilities. In addition, the Target Technical Performance measures are compared at the bottom of the section. Figure 7 shows the target section for the CHIPSat.

Objective Measurements	Measurement units	# of accel	# gages	W	m/s	$	kg	bites	b/s	$	years	GPS or not	meters
	CHIPSAT	3	2	5	7*10^7	0	0.004	3*10^2	5.2*10^3	5*10^3	2	1	4*10^16
	Mars Rover	3	463	1000		0	1025	8*10^6	2*10^6	2.5*10^9	5	1	4.1*10^7
	Suomi	3	48	370		0	435	9.2*10^6	3.7*10^4	8.1*10^9	8	1	8.2*10^5
	ESTCube-1	0	3	3.6		3*10^3	1.048	4*10^3	1.4*10^2	1.5*10^5	3	1	6.7*10^5
	Solar Orbiter	3	325	3830		0	209	3*10^5	9.7*10^5	4.5*10^8	4	1	7.5*10^10
	Bangabandhu	3	324	288	3.*10^3	2.7*10^5		7*10^4	4.6*10^5	2.48*10^8	5	1	3.6*10^7
Total Difficultly (1 - Low, 5 - High)		2	2	4	5	1	4	2	2	1	2	1	5
Imputed Importance		0.0675	0.33	0.5	0.2	0.55	0.3	0	1.15	1.1	0.4	1.2	0.2
Estimated cost (1 - Low, 5 - High)		2	2	5	5	1	3	3	3	1	2	1	5
Targets		0.0675	0.33	0.5	0.2	-0.55	-0.3	0	-1.15	0.9	0.2	1.2	0.2

Figure 7: The Target section of the QFD. This chart compares metrics of competitor products to that of the CHIPSat.

The Engineering Characteristics are also analyzed to find correlations in the Interrelationships Matrix. This matrix is useful to recognise any characteristics that have the potential to be affected by one another. Direct relationships are marked in the matrix with a check mark and inverse relationships are marked with an “X”. A strong relationship may be recorded with a double check or double “X”. The result is a lower off-diagonal matrix. An example can be seen below in Figure 8.

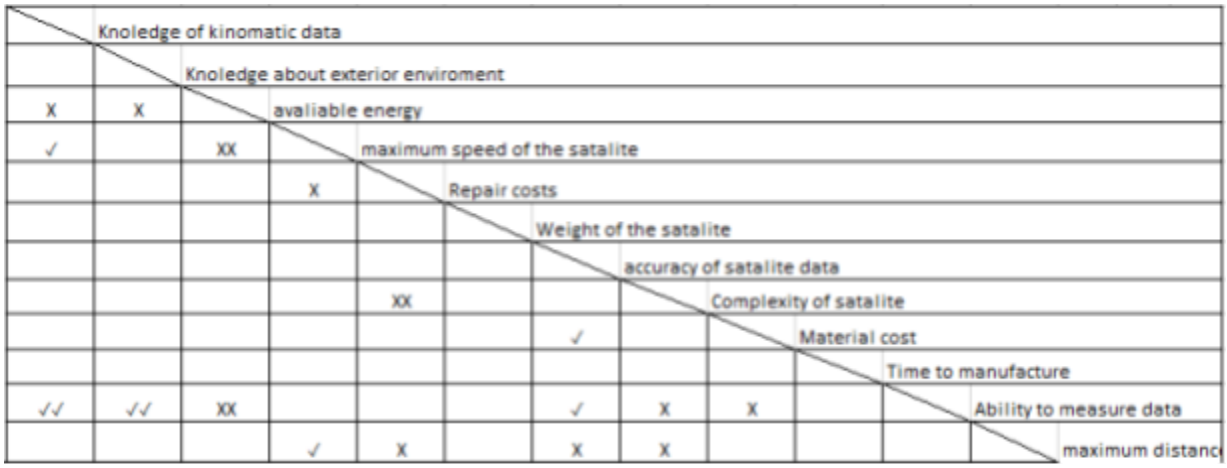


Figure 8: The Target section of the QFD. This chart compares metrics of competitor products to that of the CHIPSat.

Finally the center of the QFD can be filled out. This is done in accordance to satisfy the Impact of Engineering Characteristics on Customer Objectives. This matrix ties all of the other submatrices together in the QFD. Similar to the Interrelationships Matrix, checks and “X”s are used to reflect the relations.

Building the QFD model, serves to prioritise the engineering characteristics in the way that they are perceived by the customer. However the QFD is able to accomplish so much more. The QFD organizes the Engineering Characteristics of the product based on how they relate to each and to the competitor’s products. In addition, it shows the engineer the product of interest through the eyes of the consumer and compares that to the consumer's perception of a competitor’s.

This process distinguished the CHIPSat among the competitor’s satellites based on its engineering characteristics. However, the QFD revealed the overlooked customer perception of the project. According to the Decision Matrix, it was evident that customers were not well informed about the Alpha mission’s capabilities. This highlighted a need for customer education. Videos were made to attract interest to the Alpha project.

Sub-System Definition & Allocation

A preliminary tool in the modeling of interfaces is the Subsystem Definition and Allocation Matrix. This matrix is designed to split the system requirements and their correlating components into the corresponding subsystems. This is done by labeling the

subsystems at the top of each column of the matrix. The requirements of the subsystem can then be written in the rows below. Under each requirement, the components that are responsible for meeting the requirement are listed. This produces a well organized matrix split up by the subsystem and categorized by the requirement. This can be seen in Figure 9 below. Filling out the Subsystem Definition and Allocation Matrix for the CHIPSat was an effective strategy in compartmentalizing the overall system into the subcomponents and their respective requirements.

Navigation system	Measuring sytem	Transmitting system
The system shall track its position using a GPS system	**The system shall be able to measure pressure**	**The system shall be able to recive commands electromagnetically**
GPS	Barometer	Electromagetic reciver
The system shall be able to track its own movement using three accelorometers	Microcontroller	**The system shall be able to make packets of data and code such that the accuracy can be measured**
X-axis accelorometer	**The system shall be able to measure the temperature**	Code composer studio
Y-axis accelorometer	Thermocouple	TI-RTOS (Texas Instaments Real Time Operating System)
X-axis accelorometer	Microcontroller	DPC (Data Packet Compiler)
		The system shall be able to transmit the data recorded
		Electromagnetic transciver

Figure 9: A segment from the CHIPSat Sub-System Definition and Allocation Matrix. This matrix is organized by the subsystem and categorized by the requirement

Operations Description Template (ODT)

Another crucial diagram used in Model Based Systems Engineering (MBSE) is Operations Definition Template (ODT). This diagram is structured very similarly in nature to the Use Case Behavioral Diagram. As in Use Case Behavioral Diagram, the subsystems are listed at the top of each column. However, the ODT is generally constructed to encompass the system at large instead of just a subsystem. In addition, the state of the system and the timeline are recorded in the rightmost columns.

However, the defining factor of the ODT is the interface rows inserted during a state change. A state change should occur between column changes. The only instance where this does not occur is in a stable state, where the change does not have a correspondence with the previous entry. Each interface row is created by adding an abbreviated name and type to define the interface in the column corresponding to the last subsystem. Additionally, the color is commonly changed such that these rows are easily differentiated. This is important because, at times, the ODT breaks away from the otherwise chronological timeline with the incorporation of logical functions. This allows “if”, ”for”, or “while” loops to skip backwards or forwards to an interface. This can be seen in Figure 10, a snapshot of the CHIPSat ODT.

Opperator - Philip	Processor	Thermocouple	ADC (Analog to digital Converter)	Tranciver	State	CHIPSat progression
	Info. Event ("Queuing ")					
	If a Semaphore post is recived: processor queues the thermocouple Else: Go to ("Queuing")				Queued	Processor
	A pulse is sent from the processor to the thermocouple					

Figure 10: A snapshot of the CHIPSat ODT. Here a logical function can be seen in the in the second row.

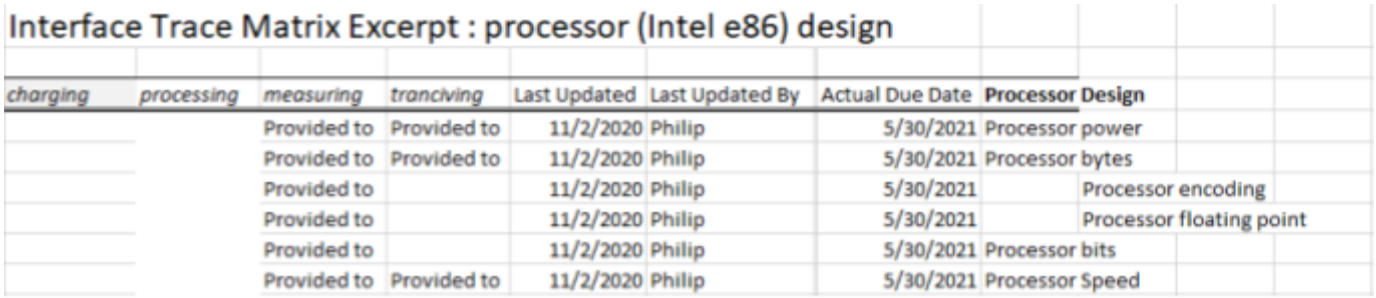
Creating the ODT was beneficial in determining the interfaces. It created a simple structure with the CHIPSat timeline and logic together in one diagram. This proved important in the modeling of queueing theory. The Queue was much easier to analyze using an ODT because of the freedom to combine both chronological and logic based progressions. Using the ODT prompted the creation of further logic maps to continue building in this area. In addition, by having clearly defined the interfaces, both the state and interface matrices were easily generated.

State Diagram & Matrix

To track the states of the system and label the correlation between the states, a State Matrix is created. A state matrix takes all of the states from the ODT and lists them chronologically across both the rows and the columns.The interfaces between the states are filled in to the entries of the matrix. Each matrix entry specifies the interface of the row state with the column state. However, the interactions are not necessarily reversible between states. This means that the resultant matrix is not necessarily symmetric. In addition, since a state can not interact with itself, the diagonal entries are left blank. Using the State Matrix, therefore, was insightful in determining which of the states would be reversible. The assumption that all states are reversible is a common misconception. For the CHIPSat, this mistake was made early on, assuming that if the system could transmit data it could also receive data. Although it was a stupid mistake, it could have become a detramental error. Luckily, this mistake was caught early on through testing. However, this would have been easily caught by the State Matrix.

Interface Matrix

Another byproduct of the ODT is the Interface Matrix. The name Interface Matrix is used interchangeably to refer to two different models. The first one consists of several interrelated tables, each made for a different subsystem of the ODT. The task entries in the respective subsystem column of the ODT are recorded in the table. The Interface Matrix elaborates on each of these task entries, adding several columns for the due date, last updated date, updated personnel, value, and units. Additional columns exist for each of the other subsystems such that the interface of each task entry can be recorded (see Figure 11).



Interface Trace Matrix Excerpt : processor (Intel e86) design

charging	*processing*	*measuring*	*tranciving*	Last Updated	Last Updated By	Actual Due Date	**Processor Design**	
		Provided to	Provided to	11/2/2020	Philip	5/30/2021	Processor	power
		Provided to	Provided to	11/2/2020	Philip	5/30/2021	Processor	bytes
		Provided to		11/2/2020	Philip	5/30/2021		Processor encoding
		Provided to		11/2/2020	Philip	5/30/2021		Processor floating point
		Provided to		11/2/2020	Philip	5/30/2021	Processor	bits
		Provided to	Provided to	11/2/2020	Philip	5/30/2021	Processor	Speed

Figure 11: A snapshot of the CHIPSat Interface Matrix. Each task entry interface can be tracked to other subsystems as seen in the left hand columns

The Interface Matrix is the best systems engineering model to track any of the interfaces between parts. Having used this model, it became abundantly clear what interactions had to be made on the software side of the CHIPSat modeling. This model defined the interaction between each element as listed in the ODT. The second Interface Matrix can be made by condensing these tables. This Interface Matrix is extremely simple to make, as the format is identical to that of the State Matrix. The only difference is that the subsystems replace the states, and the interfaces fill the matrix.

Sequence Diagram

The Sequence Diagram is a similar, more visual interface model. The Sequence Diagram is structured very similarly to the ODT, however, it is more like a block diagram in nature. Like the ODT, the subsystems are represented at the top of the page, and the diagram flows from top to bottom. A timeline is made that weaves back and forth though the different subsystems. Because of this structure, the sequence diagram offers a very precise time dependent ordering of events. In addition, there is a lot of versatility in representation for this diagram. For example, solid lines track actions, dotted lines track communication, and loops or logical functions define complex sequences. These two characteristics, precise time structure and versatility of representation, were extremely beneficial to CHIPSat modeling efforts due to the clarity of the communication between the processors and other subsystems. Additional modeling efforts were made to write pseudocode for how those communication channels would interface. In the end, the Sequence Diagram was the most representative of the utilization lifecycle of the CHIPSat.

Behavioral Test Plan/Test Methodologies

Testing the CHIPSat spanned beyond the Goal Question Metric (GQM). Before the CHIPSat was advanced to the production phase, all of the requirements had to be verified. This could be done by creating one of two types of tests. The first tests the behavior of the system and is appropriately referred to as a Behavioral Test Plan. A Behavioral Test Plan table is generated with the test ID, test method, test facilities, and

the entry and exit conditions. This table makes it easy to refer back with the test number and recount certain behavioral characteristics of the system.

The second table is used for testing qualities of the system and is called the Test Methodologies table. This table is similar to the behavioral test plan in structure, but content-wise is focused on meeting the requirements rather than noting a behavior. Therefore, a requirements column is added to note a defined standard for the test. This is paired with the originating requirement that supported this standard. In addition, the method of verification is clarified as well. The system can be tested using analysis, inspection, demonstration, or a physical test.

Keeping the test data in simple tables like this made it easy to refer back to tests. When programming the CHIPSat, this became very useful because of the constant need to update the code. Tests would often have to be redone or behaviors noted for corrections on the software side. In addition, looking back at the requirements, each one could be verified by following the tests in these tables. Verifying that each requirement had a corresponding test was done in the Verification Cross Reference Matrix (VCRM).

Verification Cross Reference Matrix (VCRM)

To elaborate, VCRM is made to verify that each originating requirement is paired with at least one test procedure. This matrix is set up with the originating requirements listed along the columns and the test procedures along the rows. An “X” is placed in the matrix if the test procedure verifies the originating requirement. If there is not a test procedure for each requirement, new test procedures will have to be added to either the Test Methodologies or the Behavioral Test Plan. This matrix serves to verify that testing strategies exist for each requirement. For the CHIPSat project, this was insightful because many of the test plans had not been explicitly recorded. When writing the software, many times the requirements are able to be checked simply by building or uploading the code. Because of this, some of the test plans had not been properly recorded, and some had been missed altogether. The VCRM pointed out those shortcomings, and they were able to be fixed in our verification plan.

Severity Rating System & Likelihood Rating System

Just because a system is able to pass all of the test procedures does not mean that it works as intended and is ready for market release. The risk of failure for the failure modes must be evaluated such that higher risks can be mitigated. There are two metrics that comprise risk: the likelihood of failure, and the severity of the repercussions, if failure were to occur. In order to evaluate the failure modes, a Severity Rating System and Likelihood Rating System must be developed. This can be done by rescaling the likelihood and severity of a failure on a scale from one to five or one to ten. It is important that the scale is clearly defined.

Risk Priority Number Table

The next step in assessing risk is making a grid and creating a Risk Priority Number Table. The grid is made such that the severity is plotted on the x-axis and the likelihood is plotted on the y-axis. Each spot in the grid is given a score based on the multiple of the likelihood and the severity. To identify high and low risk regions, a rating system is created where the highest risk bracket is colored red on the grid and generally spans a risk factor of fifteen to twenty five. The middle region of the grid is typically colored yellow and spans a risk factor of five to fourteen. The lowest risk region exists with a risk factor under five and is colored green. The resultant Risk Priority Number Table should resemble Figure 12 below.

Risk Priority Number (RPN) Definition Table

Likelihood							**Risk Criticality Ranges**	
5	5	10	15	20	25		15-25	High Risk
4	4	8	12	16	20		9-14	Medium High Risk
3	3	6	9	12	15		5-8	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*		

Figure 12: The Risk Priority Definition Table is shown above. Using the severity and likelihood failure coefficients the Risk Criticality can be evaluated

Failure Mode and Effect Analysis (FMEA)

Finally, the risks found in the Risk Priority Number Table is used to fill out the Failure Modes and Analysis Evaluation (FMEA) table. This table dissects subsystems into possible modes of failure. For each failure mode, a possible failure cause is hypothesized for its respective effect. The failure mode is rated based on criteria from the Risk Priority number table (i.e. the severity, likelihood, risk priority number, and risk criticality). If the risk is deemed low, no further action is required. However if the risk is rated high, a corrective action is implemented, and a new iteration of the FMEA is required. This iteration continues until the risk is averted. The FMEA was not the primary strategy used for the CHIPSat, but it verified that all of the mission risks were kept low. Of course, with a proof of concept space mission there is always risk, and therefore every failure mode should be tested. Luckily, until deployment the software itself has a low severity if something were to go wrong. This allows testing to carry into even the production phase without significant financial repercussions.

Event Tree/Fault Tree

A way to calculate risk quantitatively is by using an Event Tree or a Fault Tree. These diagrams are very familiar across the sciences, the most well known of which may be the family tree from biology. Both trees start at a node, and branch out after each new event. These trees can be applied to risk in Model Based System Engineering. A Fault Tree is constructed by starting from the undesirable event and working towards the causes. An Event Tree is structured vice versa, starting with one of the initiating events and working forward in time from the cause to the undesirable end events. Figure 13 shows an example of an Event Tree used in the CHIPSat risk assessment.

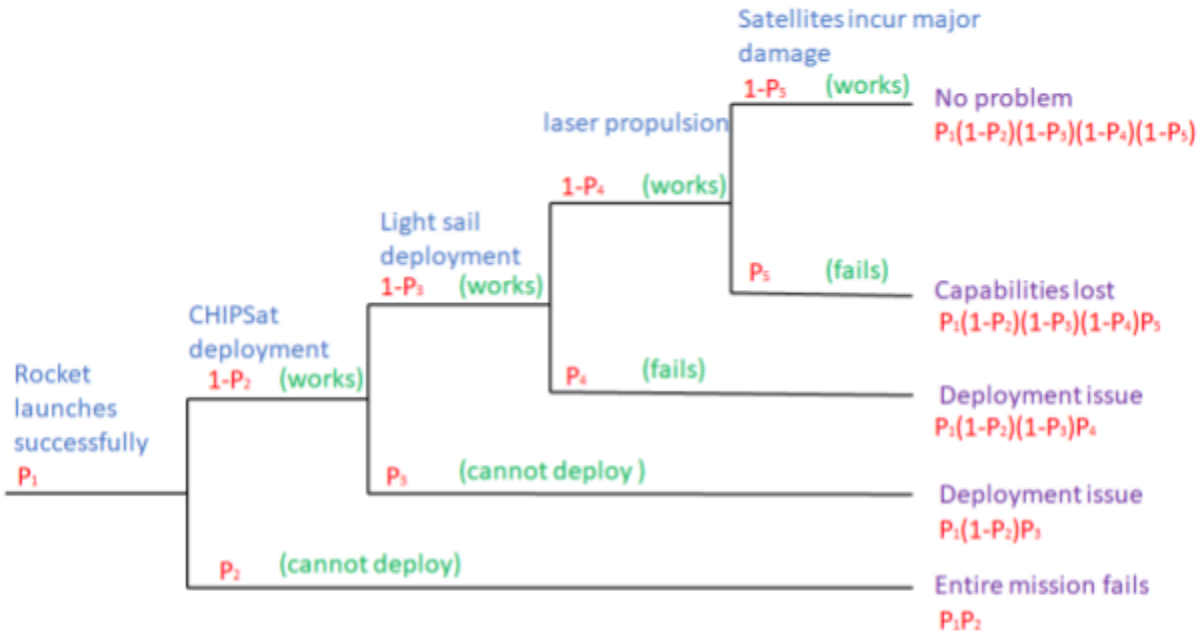


Figure 13: The event tree above shows the possible undesired events. The probabilities can be calculated using simple statistics

The probabilities of each event can be calculated using simple statistics. The Event Tree was more difficult to incorporate in the CHIPSat risk analysis. This was due to the fact that the failure probabilities are not explicitly known. Discovering these probabilities could be done by revising the GQM.

Parametric Diagram

When modeling mathematical relationships, a useful block diagram with quantitative representation is the Parametric Diagram. This block diagram follows a flow chart model, starting with all of the inputs and mathematically progressing towards the outputs of the system. For each step, the variables are stored in the boxes and a mathematical relationship is established, depicted by a binding connector. This can be seen in figure 14 below.

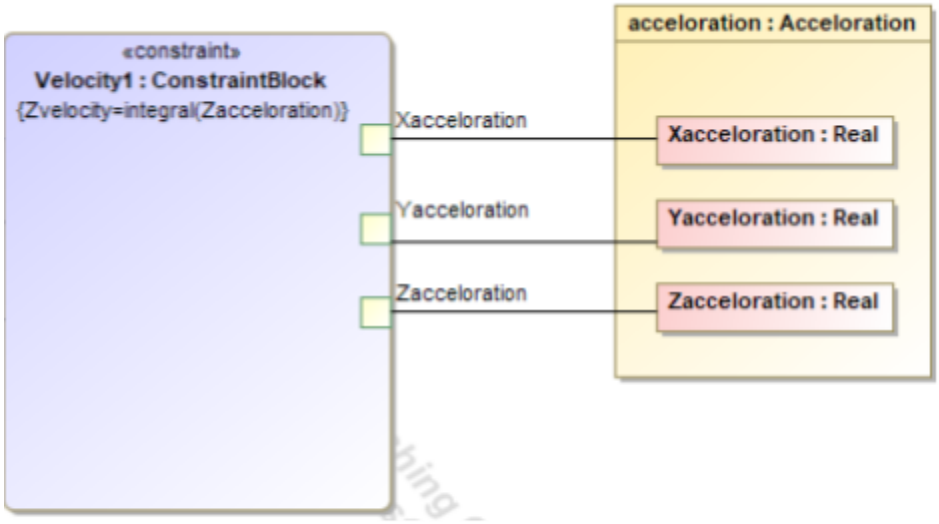


Figure 14: The Parametric diagram is used to show mathematical relations between elements in the model

For Alpha, this diagram was used in modeling the calculation of the kinematic properties of the CHIPSat. Since the system used a six axis accelerometer, the acceleration could be integrated with respect to the CHIPSats internal clock to find velocity and displacement. Because the system is not constrained in space in any way, for displacement, velocity and acceleration, there are six degrees of freedom each. Using the Parametric Diagram helped validate the code, by providing a clear and outline visual representation of all of the relations. In this setting, it was much easier to catch and correct any errors.

Timeline GANTT chart

Finally, the last model used was the timeline. A GANTT chart was developed for this function. This chart lists all of the system requirements along the rows, and a timescale is given by the columns. Each requirement is further broken down into manageable step-by-step tasks. These tasks have the ability to be finished individually, and are broken down so that they are not overwhelming. Each task is assigned to a team member who is responsible for completing it by a deadline.

For the Alpha project, a GANTT chart was created both for the entire team and for my work on the CHIPSat specifically. When working as a team, teammates are often

reliant on each other. For the team it was beneficial to have a general timeline, such that students could check in on eachother and hold other students accountable. For an individual GANTT chart, it was beneficial to split tasks up so that no specific task felt too overwhelming. Tackling manageable tasks at a time, made the responsibilities clear and increased productivity.

In conclusion, using MBSE techniques in developing software for the CHIPSat, increased the reliability of the system. It is easy to overlook facets of a project when many people with a diverse set of responsibilities are involved. These models bring clarity to the system needs and lower the chance of mistakes occurring due to miscommunication. Following the MBSE structure rigorously assures the satisfaction of the engineer, stakeholder and customer. In addition, the clear documentation of the project allows for further development.

References:

INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

Schneider, David R. “Model Based Systems Engineering Lectures 1-22.” SYSEN 5100. Cornell University , 2020, Ithaca, New York .