**Cleaning Up Low Earth Orbit:**

**A System Design Application for Space Debris Removal**

Taylor Ribeiro, [tmackinnon@gwu.edu](mailto:tmackinnon@gwu.edu)

Taylor Gregory, [taylorgregory@gwu.edu](mailto:taylorgregory@gwu.edu)

Davis Huffman, [davishuffman@gwu.edu](mailto:davishuffman@gwu.edu)

EMSE 6099: Problems in Engineering Management and Systems Engineering (Fall 2 2022)

Dr. Shahryar Sarkani

December 15, 2022

**Abstract**

As satellite and spacecraft technology has advanced over the past several decades, the amount of space debris in low earth orbit (LEO) has increased exponentially. This debris has narrowed windows available to launch new satellites and other spacecraft. If the issue is not addressed, we could eventually lose the ability to execute new launches all together. Many different space agency groups have come up with ideas and proposals for how to combat this issue. However, since this is a worldwide issue, it is important that information on how best to execute this is shared between organizations. To promote this information sharing, a software-based decision-making tool will be developed to store data from all organizations working on this issue. The success and failures of all missions will be analyzed to give future missions the best chance of success. Other factors such as budget and schedule will also be considered to always give organizations the best chance of success with the resources that they have available. Several evidence-based artifacts have been designed and developed to describe this software system. Exploring all aspects of this system, artifacts include overviews of intended use and stakeholders, details on decision making within the software, and cost and schedule analyses to complete and upkeep this system. This system will address worldwide implications of the space debris issue that affect most people on earth at some level.

* **Problem Statement**: There are over 27,000 pieces of space debris orbiting earth resulting in hazardous conditions for human space flight and robotic missions. (21)
* **Thesis Statement:**  A software-based decision-making tool will be needed to aid engineers in designing a system to reduce space debris, resulting in decreased satellite loss and hazards for manned flight missions. (29)

**Introduction**

There are over 27,000 pieces of space debris orbiting earth resulting in hazardous conditions for human space flight and robotic missions (Garcia, 2015). Space debris is a product of dead satellites, used rocket stages, and fragments from the collision of these components (Mark, 2019). This is a huge risk for both human space flight, including any spacecraft manned by humans traveling in low earth orbit and robotic missions, including that of satellites and other unmanned crafts. The International Space Station regularly tweaks its orbit to avoid potential space collisions with debris (David, 2021). Space debris has a negative impact on Aerospace communications impact geo-location, prominently used by government, military, and the average public space. If space debris continues to accumulate at its current exponential rate, the safety of human space flight will diminish further and our ability to use geo-location will become even more difficult.

Removal of the space debris can be achieved by deploying a space debris cleanup system. This is a problem that has been thoroughly researched and numerous solutions have been developed in the scientific community over the last 50 years. As a result of years of this research, there are about 8 types of Active Debris Removal (ADR) Methods: collective, laser-based, ion beam shepherd-based, tether-based, sail-based, satellite-based, dynamical systems-based, and unconventional (Mark, 2019).

It would be beneficial for engineers to be able to tap into this historical research and development as they work on developing new space cleanup systems. Utilizing previously researched insight will save engineers precious time and money and help them to solve the space debris issue once and for all. To achieve this, a software-based decision-making tool will be needed, which considers historical research and development in space debris cleanup systems. The decision-making tool is powered by a database of internal research that takes inputs and queries from the end-user and provides a recommended space cleanup system design solution. The database is regularly populated with data from prior research and development on space debris cleanup systems.

This software-based decision-making tool would be utilized by aerospace agencies (NASA, ESA, etc.) when they develop systems to clean LEO (low-earth orbit) hazardous space debris. The software will be marketed as a Software as a Service Solution (SaaS), for all users of the scientific community. Space debris is a serious issue impacting all of earth’s LEO, and organizations across many different countries are keeping a close eye on the issue (Duta, 2022). Therefore, this is a software tool that will be available to all scientific organizations across the world as we jointly combat the issue.

**Table of Supporting Artifacts**:

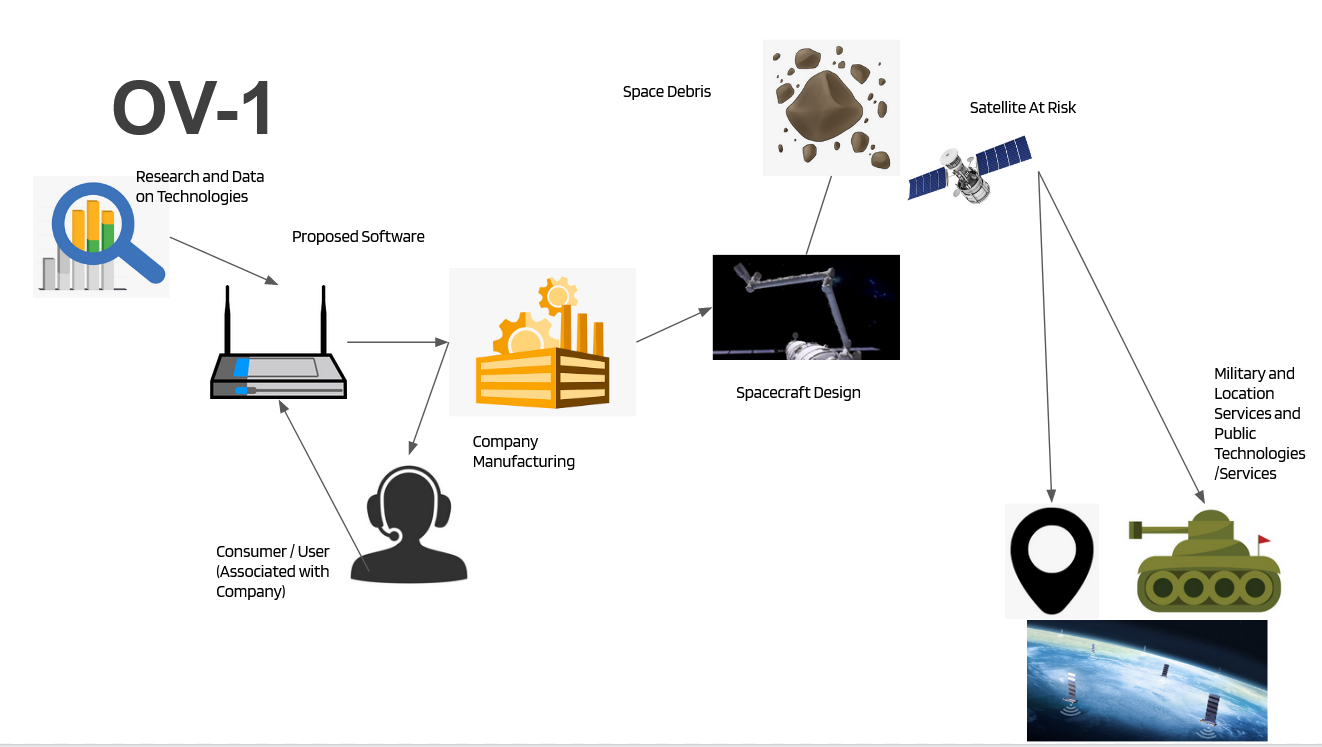
|  |  |  |
| --- | --- | --- |
| **Figure** | **Artifact** | **Course** |
| **1** | OV-1 | EMSE 6840 – Applied Enterprise Systems Engineering |
| **2** | Functional Decomposition | EMSE 6801- Systems Engineering 1 |
| ***3*** | Requirements | EMSE 6815 – Requirements Engineering |
| **4** | Screen Mockups | EMSE 6805- Systems Engineering 2 |
| ***5*** | Decision Tree | EMSE 6805- Systems Engineering 2 |
| **6** | Software Pathing Diagram | EMSE 6817- Model Based Systems Engineering |
| **7** | ERD Diagram | EMSE 6817- Model Based Systems Engineering |
| **8a, 8b** | Work Breakdown Structure (WBS) | EMSE 6805- Systems Engineering 2 |
| **9** | Schedule | EMSE 6825- Project Cost & Quality Management |
| **10** | Stakeholder Analysis | EMSE 6825- Project Cost & Quality Management |
| **11** | SWOT Analysis | EMSE 6805- Systems Engineering 2 |
| **12** | Break Even Analysis | EMSE 6410- Survey of Economics and |

**OV-1 Diagram**

DoDAF, or the department of defense architectural framework is a breakdown of models and diagrams that allow for a common approach in presenting architectures across organizational boundaries. The premise is to use common terminologies and assumptions to present information to all stakeholders in an enterprise. DoDAF is broken down into viewpoints and views which target domains within an industry. DoDAF divides viewpoints into sections such as ‘All, Data and Information, Standards, Capabilities, Operational, Services, Systems, and Project.’ Each of these viewpoints have views specific to aspects of the viewpoint to relay certain pieces of information. For this project, homing in on the operational view for general stakeholder awareness was a prime choice.

The Operational Viewpoint describes the operational elements and flow exchanges in order to conduct operations (United States DoD, 2022). Think of it as a general understanding of how the system will operate. A significant benefit from the operational view is that it helps identify boundaries in the high-level system. Boundary definition uses the models from the operational view and incorporates stakeholders to home in on where ideal support should be allocated, allowing determination of necessary interfaces for organizations associated with the system. This analysis, even at a high-level, creates functional scope and organizational span.

The OV-1 view describes a mission, class of mission, or scenario. It shows the main operational concepts and interesting or unique aspects of operations. The OV-1 provides a graphical depiction of what the architecture is about, and the players and organizations involved. Its main use is to aid human communication and is intended for presentation to high-level decision makers. For this project, the OV1 can be shown below:

****

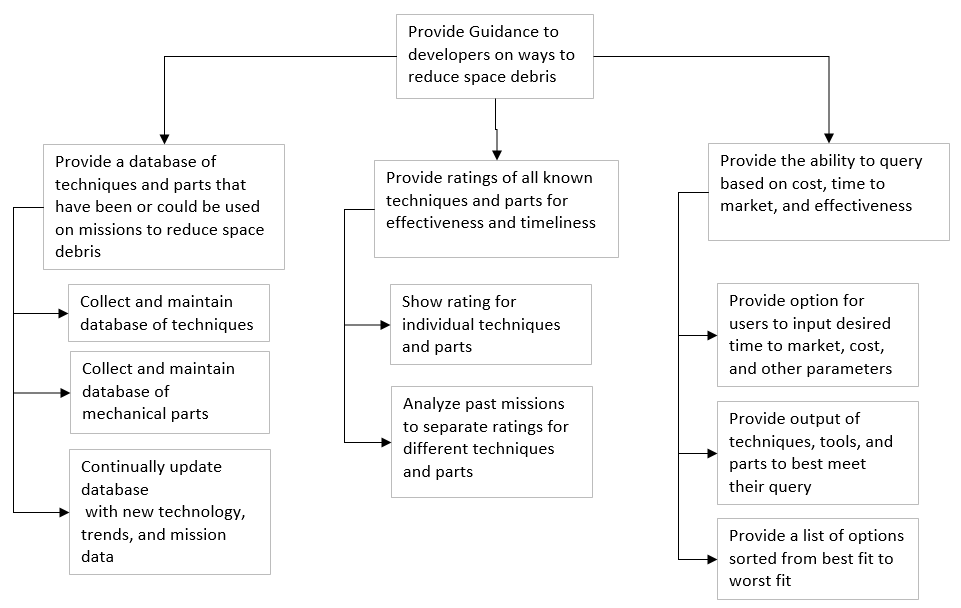
**Figure 1** *- OV-1 Diagram*

This OV-1 highlights how our software system will interact with every aspect of the architecture and the stakeholders interact. Our proposed software, and product, will collect research data and technologies associated with spacecraft creation along. This information collected is most of the benefit that the software provides, allowing the user to bypass their own research and save the consumer time. The software will query the user and scour its internal database, providing an estimated system based on the needs and constraints of the user. From there, the user has the ability to choose a proposed debris elimination system and get right to production. The consumer would move to the manufacturing stage, external to our system, and create their proposed system. The spacecraft design would progress into a working product that is then sent to eliminate harmful debris in Low-Earth Orbit. After the success of the manufactured system, military systems, location services, and public technologies will be positively affected by the lack of damage to the spacecraft counterparts. The aforementioned technologies, such as GPS, are highly reliant on successful communications with satellites in Low-Earth Orbit and can continue to function for the foreseeable future based on the solutions the software is providing.

**Functional Decomposition**

The functional decomposition of this system shows a high-level goal and breaks it down into sub functions which can then be implemented as features in the software system. Functionally decomposing an idea is a systematic process that helps developers better understand their system and scope of work. It is also a great tool to communicate with stakeholders to ensure that the functionality that they expect is present in the system. The functional decomposition should encompass all of the intended functions and show how they will relate to each other. Figure2 shows the functional decomposition of the software-based decision-making tool for the development of a space debris clean up system.

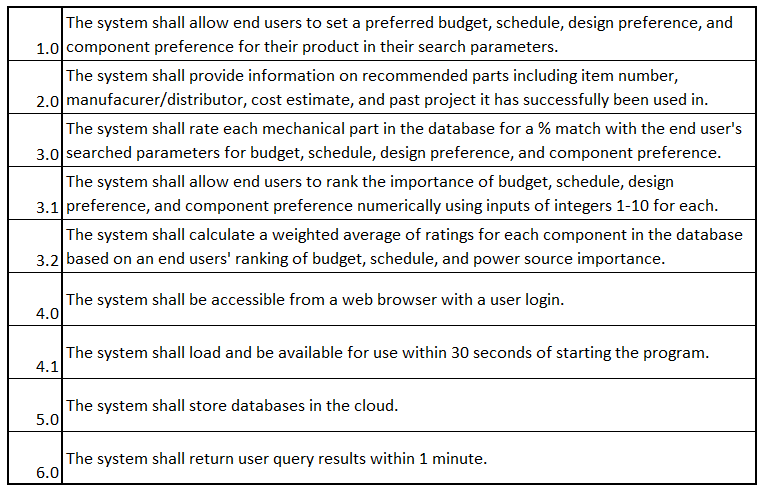
The high-level functional goal of this system is to provide guidance to developers who are working on creating a system that will help to clean up debris in low earth orbit. This is decomposed into sub-functions. The first sub-function is to provide a database of parts and techniques that can be used to develop a space debris clean up system. This is further decomposed into collecting and maintaining information for these databases and continually updating these to stay current with the latest information. The second sub-function is rating all of the techniques and parts for effectiveness. In this case, effectiveness is referring to the ability of the part to meet criteria such as budget and schedule set by the user. This is further decomposed into analyzing data from past missions to collect this data and show the ratings calculated from that data to the end user upon query. The final sub-function provides the end user the ability to query the databases based on total system cost, time to market, and other effectiveness parameters. This final sub-function is broken down into providing end users input boxes to present this data and providing an output that meets their criteria listed from best fit to worst fit.

**

**Figure 2 -** *Functional Decomposition Diagram*

**Requirements**

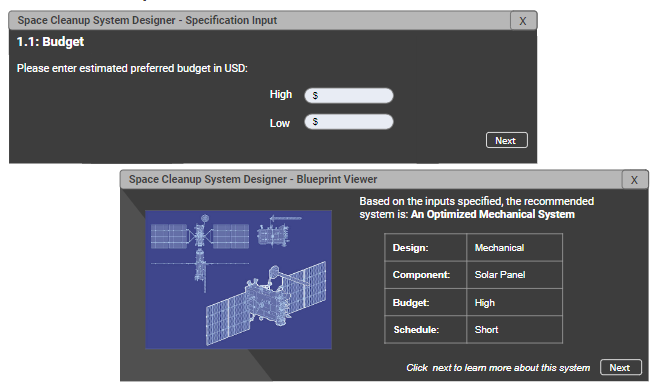
Asystem requirement must have several characteristics to be effective including having clear, unambiguous language, making sure that it is necessary and attainable, and it should be implementation independent (Koelsch, 2016). A variety of functional and quality requirements were written for this system. The functional decomposition gave direction to the writing of these requirements, and they were tailored to a testable format. Testable format requires all the characteristics above so that there can only be one interpretation of whether a requirement passes or fails its test. With requirements written this way, there is far less frustration or debate towards the end of the project on whether they have been met. The requirements listed in Figure 3 are a translation of user needs into a format that both the end user and the developers can understand.



**Figure 3 -** *Requirements*

**Screen Mockups**

Screen Mockups are typically generated during the design phase of a software project, to help the development team gain an understanding of what the finished product may look like. Having this done during the design phase helps mitigate rework done later in the development or post-development phases, which can be extremely costly. Screen Mockups are usually developed based on requirements provided by the customer/end-user in conjunction with the development team. They are a good tool in identifying any missed requirements, or user preferences prior to the actual development.

**

**Figure 4 -** *Screen Mockups*

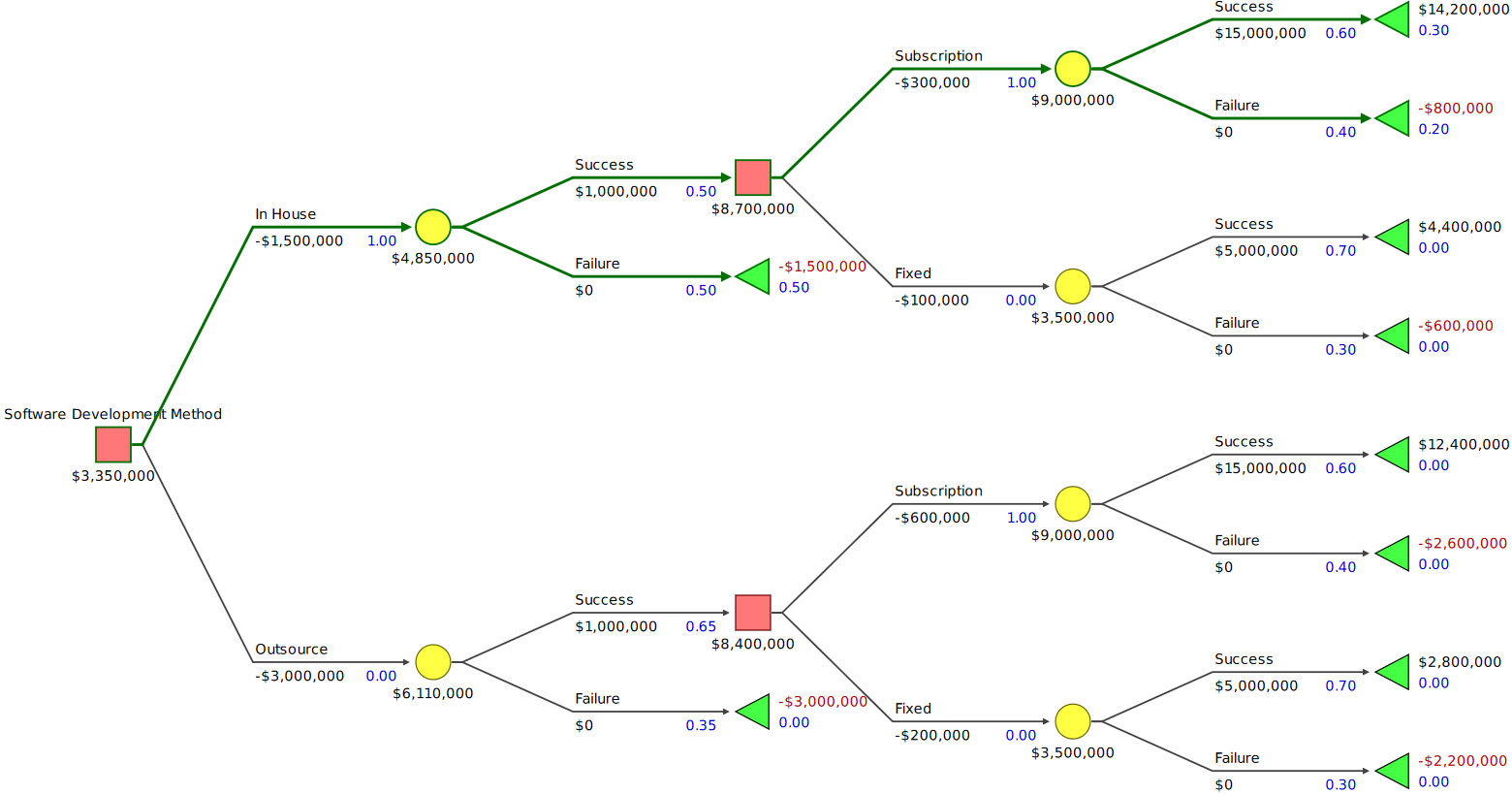
Figure 4 shows a couple of the most important screens of the Space Cleanup System Design Tool. The specification input window is the screen the end-user will input their desired specifications of their system and the project. This includes inputs like their project budget (shown), as well as system components shown in the requirements. Once all inputs are complete, the system will generate the system design solution, and the BluePrint Viewer window will pop up to display the result. This window shows a snapshot of what the system design solution contains. The end-user will then move to the next screen (not shown) to get a more detailed output of the system and the suggested design.

The Screen Mockups for the Space Cleanup System Designer are developed by the Software and Systems Engineers during the design phase and will be presented to a focus group consisting of members from NASA to obtain user feedback. All user feedback is addressed at this stage and will prevent costly rework in the developed software system. And with such a tight timeline of one year of development, the team cannot afford any rework.

**Decision Tree**

The decision tree is a model that maps possible outcomes along with the probability of the events occurring. It allows the user to weigh possible actions based on their costs, probabilities, and benefits. The result should allow the user to pick the best path either informally or based on mathematical analysis. A strong advantage of a decision tree is readability of the model, and how easily they combine with other decision-making tools. Decision trees can become overly complex, so it is important to narrow down critical decisions, inputs, and objectives.

In the decision tree created for this project, the focus was primarily on the cost breakdown for creating the software and how to approach making profit based upon an initial cost. This information is crucial for stakeholders to understand the business model and follow along with the rollout process for the software. There are two primary decisions that follow software development: 1. Do we create the software in house with our team of developers, or do we hire contractors who are more experienced but will likely cost more 2. When selling the product, should the software be sold at a one-time fixed cost or be based on recurring yearly subscriptions (much like Microsoft 365). The analysis of these decisions can be shown below in Figure 5:

****

**Figure 5 -** *Decision Tree*

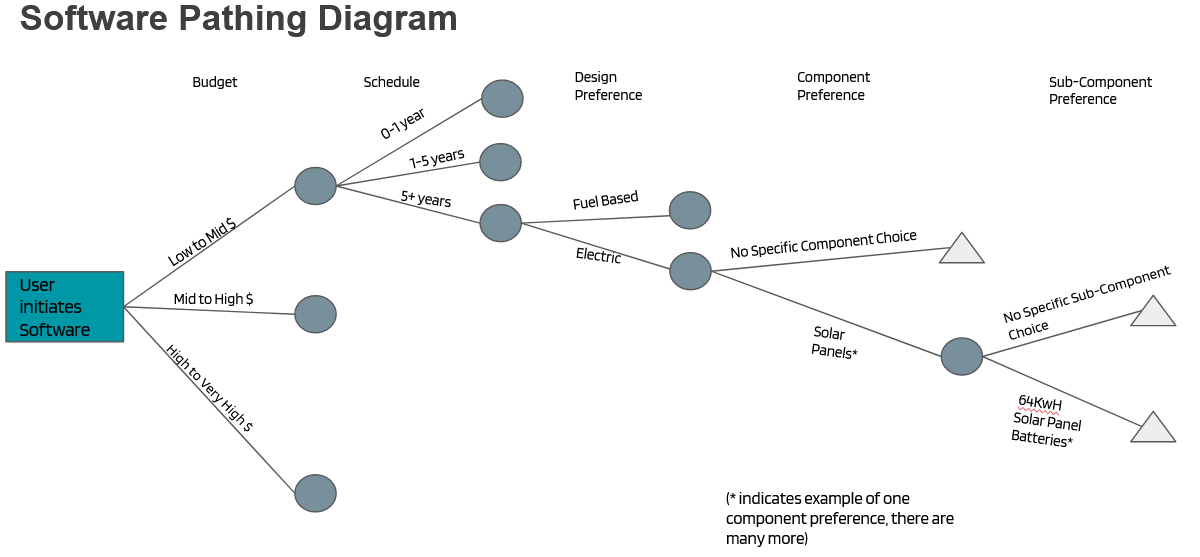
The highlighted green path shows the most optimal path in order to maximize profits from the software. The decision tree helps show that a subscription based, in house created software would result in the most profitable solution, despite outsourcing having a higher success rate. Contracted software engineers run a premium of around 2x what a normal software engineer would require for employment which when creating a fixed or subscription-based product increases the cost as well (Gmiender, 2022). Subscription based software products are ideal for projects that require a few years to complete, much like creating a space debris collection system. Since a majority of the customers or stakeholders would be operating off of DoD contracts, it is safe to assume that the licensing would be renewed for a majority (3 years) of a 5-10 year contract. (Contracts, 2022)

**Proposed Software Pathing**

For software development and software systems in general, the business stakeholders need to understand the functionality of the software. Without showing stakeholders what the system pathing would look like, there could be gaps in presenting the material to potential customers. A software pathing diagram also shows logic branches as the user navigates the system along with the subsections of criteria along the way. Since the product is assisting with scope with the end product, these criteria effectively complete the Scope, Time, Cost triangle. The criteria also give the user the ability to personalize based on their own goals and constraints.

When creating the software architecture, many factors for criteria were examined, but the final criteria selected included: Budget, Schedule, Design Preference, Component Preference, and Sub-Component Preference. Budget refers to how much cost is allocated to creating the system from the user’s organization. The database of research that is behind the software can then eliminate technologies out of budget. Schedule is analyzed on a timeline basis and refers to the availability of parts and technologies, i.e. a prototype battery may be stellar in performance but would not be able to be manufactured by the time the user’s project needs to complete. Design preference refers to the larger aspects of the system, such as the decision to use a fuel or electric propulsion system. This assists with narrowing down systems that are not remotely viable for the customer’s goal. A customer could be working off of a contract that has strong environmental concerns, to which they could eliminate fuel-based propulsion and use electric to satisfy those constraints. Component preference allows for the user to use a specific technology that is desired. Within the electrical system design preference, the user could decide that they’d require solar panels because they have a manufacturing deal with another company. Sub-components are also included as certain chosen components have specificities that a user could request.

The figure below shows an example user navigating the software to an end result. It is important to note that design preference, component preference, and sub-component preference are looped. This means that the user can select multiple different design preferences or component preferences if desired. Budget and schedule do not need to be looped as those would be static inputs based on external user criteria. A more extreme example would be something such as: A high budget, over 5+ years, fuel based, mechanical arm, sun-sensors, boosters with geo-locator, titanium fuel tank, etc.



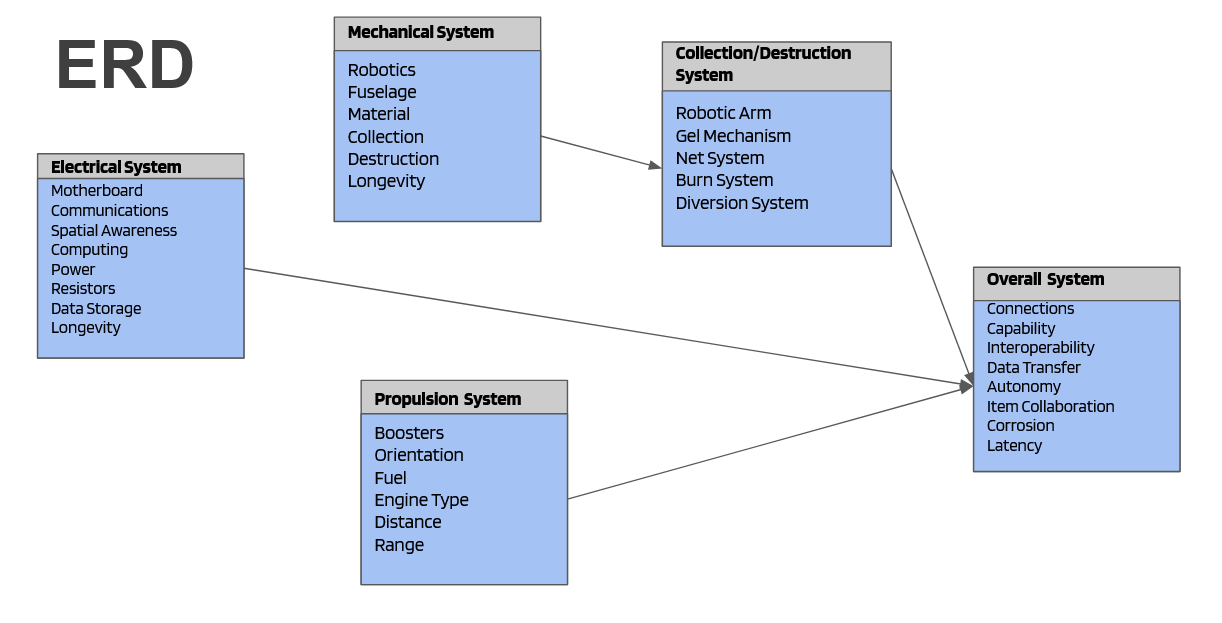
**Figure 6** *- Software Pathing Diagram*

A more extreme, but more probable, example would be something such as: A high budget, over 5+ years, fuel based, mechanical arm, sun-sensors, boosters with geo-locator, titanium fuel tank, etc.

**Entity-Relationship Diagram**

Since the deliverable for this project is a software system, it is common to see an ERD (Entity Relationship Diagram) associated with the early stages of presentation. An ERD depicts relationships between people, objects, places, concepts, or in this case, an information technology system. More importantly, ERDs are the visual starting point for database design which assists engineers on the project with knowing the requirements and aspects associated with database creation. Since the research and data collection is something, this software will be providing, that information needs to be stored in a database for the software to be able to query. Understanding how the database interacts with each system will result in a more streamlined development effort.

For this system, a conceptual data model was created, which provides an overview of the scope of the project and how data sets relate to each other. Other data models such as logical, or physical were not preferred over conceptual due to the nature of other resources in this document portraying interdependencies between aspects outside of the IT space. The conceptual ERD for this software system can be seen below:

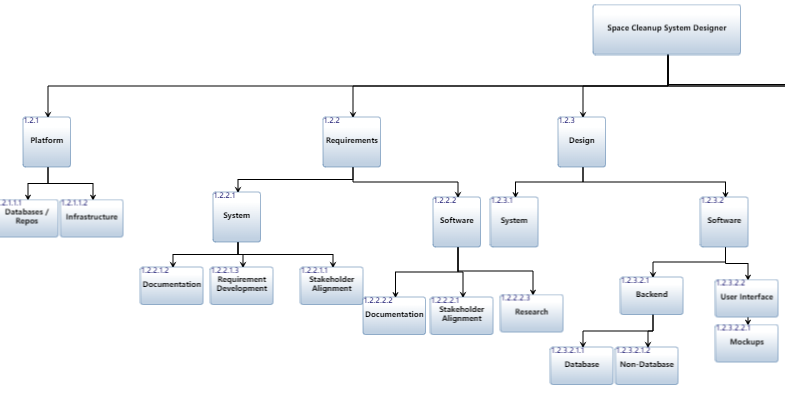


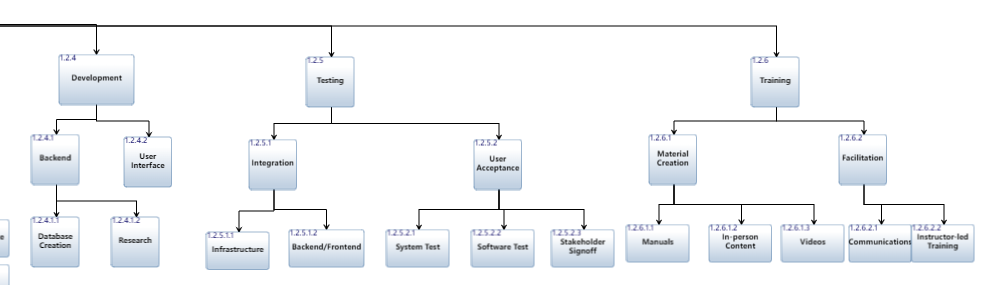
**Figure 7 -** *ERD (Entity Relationship Diagram)*

This ERD takes the four most important systems within designing a spacecraft and separates them into different buckets of information. The four main systems involved are: Mechanical, Electrical, Propulsion, and Collection/Destruction. Each system has sub-systems, representing tables of information where data will be stored. Keep in mind that this is a database design tool, so each larger system will contain tables of data for each component. An example of this would be the motherboard, found in the electrical system database, which would contain a table of different types of motherboards and the specs associated with them. The specifications associated with the component refer to user query preferences (budget, availability, performance, etc.).

When the user would input their information, the software would scour the database tables and provide potential solutions. Prior to providing the solution, the overall system must be examined. Parts found from the four main system tables must be compared with the data found in the overall system database. This rules out potential solutions that do not physically work with one another. The software could choose an electric motor and provide a fuel type that clearly doesn’t interface with an electric motor. In this case, the overall system database would weed out the item collaboration issue and disregard the suggestion prior to reaching the customer.

**Work Breakdown Structure**

A Work Breakdown Structure or WBS, is an effective way to visualize the work required to complete a project. The idea is that you take large tasks or groups of tasks and break them down into more manageable chunks that can be executed. The WBS is typically created during the project planning phase and can help with planning project costs by estimating the project’s scope and therefore resources required to complete the work. 

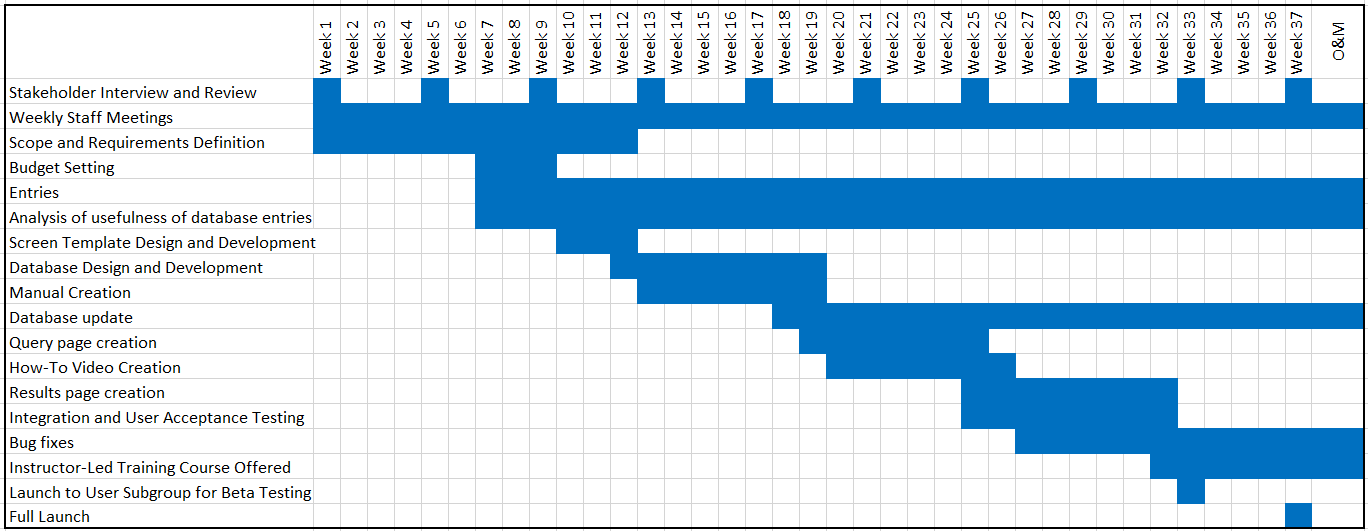
**Figure 8a -** *Work Breakdown Structure (1 of 2)*

**Figure 8b -** *Work Breakdown Structure (2 of 2)*

The work required to implement the Space Cleanup System Designer is detailed out in the work breakdown structure above. At the top level, the work is broken down into Platform, Requirements, Design, Development, Testing, and Training. As we are implementing a software-based system, this breakdown is tailored specifically for the software components required for this system, as well as supportive items like platform infrastructure and user training manuals.

This WBS is a helpful aid in visualizing the work that needs to be done so that we can identify efficiencies in our project plan. The goal would be for this project to be completed in one year, so any work that can be done in parallel should be executed as such to reduce the project schedule. Activities like design and development must be done serially. Some testing can be done in parallel with development, whereas we will have quarterly system test events to sell off requirements as the software functionality is completed. The same can be done for the training materials, being complete as individual software features are completed.

**Project Schedule**

**

**Figure 9 -** *Schedule*

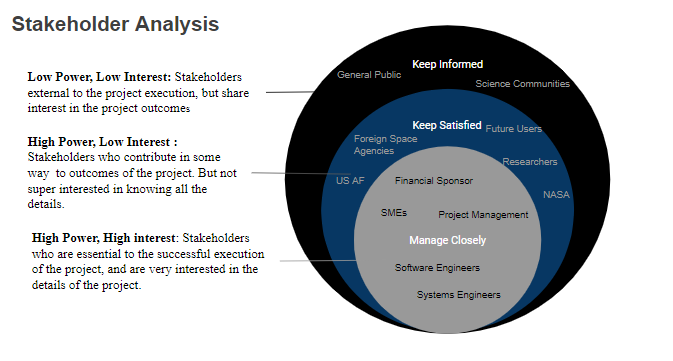
The integrated master schedule (IMS) is directly correlated to work that was defined in the WBS (Schmidt, 2013). Identifying tasks and the dependencies between them is a critical step to take before starting to assign tasks to developers or development groups. Once it is known what needs to be done and who needs to do it, the schedule can start to be drafted. It takes the eye of an experienced developer to make estimations on how long activities should take. This is a bottom-up estimation method. However, even using this method, schedules often fall behind the initial projected timelines. Therefore, significant upfront planning will be done on the development of this software-based decision-making tool to help avoid that predicament. This is a small to moderate sized software development project that involves the creation of databases and several interactive screens. These types of projects generally take 3-6 months to develop (Kytainyk, 2022). It is estimated that the software development time for this project is about 23 weeks due to this complexity. Several tasks need to be continually updated over the lifetime of this product. This is shown in Figure 10 as the operations and maintenance (O&M) phase.

During development, many unexpected issues can occur. Therefore, it is important to regularly review the schedule compared to where the project is at. Confirming that the project is on schedule is the ideal scenario for these reviews. However, if, for any reason, the project has encountered unexpected roadblocks that delay it, these schedule reviews can be used to assess the situation and develop a plan to get back on schedule or adjust the project accordingly. To account for any unexpected issues during development, there are several weeks of buffer planned in the schedule. There is a budget for a development team to be paid for a year for this reason

In a similar vein, throughout the development process, stakeholders will continually be kept informed about the status. They will have the opportunity to review plans, screens, functionality, and any other relevant information. These regular meetings will ensure that we are communicating effectively with all of our stakeholders to make the most useful product for them that we can.

**Stakeholder Analysis**

A Stakeholder Analysis is used to understand the different groups of stakeholders for a given project and how to glean insight into how to appropriately engage them to ensure their needs as well as the needs of the project are met. Different stakeholder groups have different needs in terms of the information they require and the frequency of that information. For each project, the stakeholder groups may differ drastically, but most projects will have a financial sponsor and project management groups.

**

**Figure 10 -** *Stakeholder Analysis*

In Figure 10 above, the Stakeholders have been broken up into three groups based on their level of power and interest in the success of the Space Cleanup System Designer project. Power refers to how essential the stakeholder group is to the project. Stakeholders with high power can have a tremendous impact on the project’s scope, budget, and timeline. Interest refers to how interested the stakeholder group is in the outcomes of the project. Stakeholders with high interest have a vested interest in the project outcomes, directly or indirectly. The financial sponsor, for example, is categorized as high power, high interest, because they can impact the project’s budget, and they care about the success of the project from the perspective of their own financial return on their investment. On the other end of the spectrum, the general public is categorized as low power, low interest, because they do not have any impact on the outcome of the project nor do they care about the details of the project outcomes. Nonetheless, these stakeholders in the “Keep Informed” category are included in this stakeholder analysis because they indirectly care about project outcomes because they have interest in solving the issue of space debris.

Stakeholders in each of the three buckets have different needs and therefore a communication plan must be in place to ensure their needs are met appropriately. Stakeholders in the inner ring, “Manage Closely”, must be kept abreast of updates in a timely manner. They will be invited to weekly update meetings, as well as any urgent ad hoc meetings. Those in the middle ring, “Keep Satisfied”, will be kept up to date monthly, and engaged for user feedback as needed. Those in the outer ring, “Kept Informed”, will be made aware of project outcomes when the project is complete.

**SWOT Analysis**

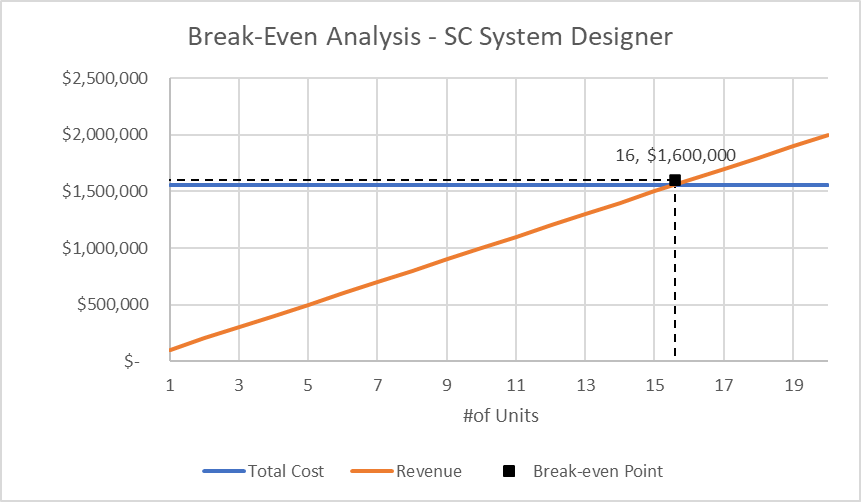
A simple SWOT analysis was performed on the project to assess feasibility as well as to help identify ways this project could grow and expand in the future. The main strength of a decision-making tool such as this project is that it will significantly decrease development time for any agency looking to help with this issue. This will drastically limit the feasibility work that needs to be done because information on successful and not successful missions has already been studied, analyzed, and consolidated for their use. A weakness of this solution is that the database needs to be updated with new data as soon as it becomes available. It is likely that some of this information may not come free to this team which could decrease profit margins. The opportunity of having this central location for knowledge will promote competition and innovation on the subject by different agencies. They will also be more likely to participate in if effort to develop a system for it is relatively low. A considerable threat to the development of this project would be that some of the information we need to make this project a success could be subject to security clearance and therefore not available for use in this publicly available tool.

|  |  |
| --- | --- |
| **Strength**  Information is consolidated from all governments and organizations involved in space debris clean up | **Weakness**  Database needs to be continually updated with latest information to remain useful |
| **Opportunity**  Encourages competition and promotes research and documentation on the subject | **Threat**  Some information could be subject to security clearance |

**Figure 11 -** *SWOT Analysis*

**Break-Even Analysis**

A Break-Even Analysis is helpful to determine the point at which a product or service will generate a profit. This is typically done early in the project life cycle before any major design or development is completed. If a project will not produce a profit, it is most likely not worth doing.

** The analysis takes into consideration the cost of generating the product or service, and the revenue that can be generated by selling that product or service. The break-even point is then calculated by dividing the fixed costs by the revenue per unit minus the variable cost per unit. This will provide the number of units required to make back the money spent on creation of the product or service.

**Figure 12 -** *Break-even Analysis*

Figure 12 above shows the Break-Even Analysis for the Space Cleanup System Designer. The cost of the project is projected to be $1.56 million, which has been estimated based on the cost of 13 full-time engineers for a 1-year project. The sales price of each one-year license of the software is priced at $100,000. Based on these inputs, the break-even point to recoup the money spent on the project is 16 units, which would result in $1.6 million in revenue.

**Conclusion**

The software proposed takes a solution space that has not been touched before in order to solve a looming problem worldwide. A surplus of harmful space debris puts satellites in Low-Earth Orbit in danger, resulting in communication loss for military, government, and public space. With the elimination of said debris, satellites that exist in Low Earth orbit can continue to function over their projected lifespans, and the window to create more satellites and advance technologically will be expanded. This expansion allows for the continual launch of satellites, improving software such as GPS due to its cluster nature (meaning the more satellites in orbit the more specific the location services can be). This also opens the door for other technologies like starlink, which uses satellites to provide internet to countries worldwide. Expansion of starlink could assist developing countries by providing cheap and reliable internet.

A software that utilizes an up-to-date and continuously updated database with physical system parts eliminates the need for additional research and can get physical systems up in orbit to eliminate debris quicker than ever before. Most projects in the spacecraft industry have a very long lead time prior to launch due to the inherent nature of research required. This software aims to eliminate that lead time and streamline production.

Profitability of the software is in the form of a licensing agreement over the span of a year and can be renewed based on the needs of the consumer. This is a stable market as government contracts are continuously being created for satellite creation and monitoring. We are assuming that the government contracts would have a budget that would be able to purchase this software for the time required, as the tradeoffs for the consumer heavily favor the purchase of the software as opposed to doing years of research and development. In sum, government contracts and private companies have the capital to be part of this software and would relieve cost and schedule implications associated with research and development.

The software timeline projects 12 months for a completed product along with continuous research for our database totaling $300,000 over each subsequent year. The software development process is composed of 13 engineers during the year of project development. After selling 16 one-year licenses at $100,000 each, the project will begin to be profitable. There is also a possibility that this decision-making framework could be adapted towards other design systems for the future, which adds to the external value of the company if the CV were ever to be sold.

All in all, the database design along with the proprietary user-friendly software will help advance technology and system designs alike. The result of this interface gives a quicker solution to the space debris problem by allowing more companies to help combat the issue for all stakeholders, regardless of their motives or specificities. A cleaner Low Earth Orbit is something our world must strive for, for the advancement of civilization. This market has already taken to the skies and has been successful, so let’s continue that trend by getting rid of debris in the world’s way.

References

Garcia, M. (2015, April 14). *Space debris and human spacecraft*. NASA. Retrieved October 26, 2022, from https://www.nasa.gov/mission\_pages/station/news/orbital\_debris.html

Davis, L. (2021, April 14). *Space Junk Removal Is Not Going Smoothly.* Scientific American. Retrieved October 26, 2022, from https://www.scientificamerican.com/article/space-junk-removal-is-not-going-smoothly/

Dutta, M., Dev, S., et al. (2022, April 28). *Extraction of Space Debris Approach: Diminish the Threats from Outer Space.* 2022 IEEE International Conference on Nanoelectronics, Nanophotonics, Nanomaterials, Nanobioscience and Nanotechnology. Retrieved December 10, 2022 from <http://dx.doi.org/10.1109/5NANO53044.2022.9828935>

Gmeinder, Coley. (2022) “Software Engineer Salary (2022).” https://angel.co/blog/software-engineer-salary.

Koelsch, G. (2016). *Requirements writing for system engineering*. Apress L. P.

Kytainyk, V. (2022, December 2). *Software Development Time Estimation: How Long Should It Take To Develop A Product?* Forbes. Retrieved December 10, 2022, from https://www.forbes.com/sites/forbesbusinesscouncil/2022/12/02/software-development-time-estimation-how-long-should-it-take-to-develop-a-product/?sh=50a0a7bf76ce

Mark, P., Kamath, S. (2019, January 29). *Review of Active Space Debris Removal Methods*. Space Policy. Retrieved December 10, 2022, from https://www.sciencedirect.com/science/article/abs/pii/S0265964618300110

Schmidt, R., & Schmidt, R. F. (2013). *Software engineering : Architecture-driven software development*. Elsevier Science & Technology.

United States Department of Defense. (2022) *U.S. Department of Defense* *Contracts*. <https://www.defense.gov/News/Contracts/>.  
  
United States Department of Defense. (2022) “1: High Level Operational Concept Graphic.” OV. https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20\_ov1/.