2022

Top-Down & Bottom-Up Approaches to Robot Design

Dylan Michael Covell
dmcovell@mix.wvu.edu

Top-Down & Bottom-Up Approaches to Robot Design

Dylan Covell

Thesis submitted
to the Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Master of Science in
Mechanical Engineering

Dr. Yu Gu, Ph.D., Chair
Dr. Jason Gross, Ph.D.
Dr. Guilherme Pereira, Ph.D.
Dr. Nicholas Szczecinski, Ph.D.
Dr. Xi Yu, Ph.D.

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia
2022

Keywords: Bottom-up, Top-down, Design Methodology, Robotics

Abstract

Top-Down & Bottom-Up Approaches to Robot Design

Dylan Covell

This thesis presents a study of different engineering design methodologies and demonstrates their effectiveness and limitations in actual robot designs. Some of these methods were blended together with focus on providing an easily interpreted project design flow while implementing more bottom-up, or feedback, elements into the design methodology. Typically design methods are learned through experience, and design taught in academia aims to shape and formalize previous experience. Usually, inexperienced engineers are taught approaches resembling the Verein Deutscher Ingenieure (VDI) 2221 process. This method presented by the Association of German Engineers in 2006 is regarded as the general system design process. This introductory process is largely left open to interpretation, and it is often unclear when to implement feedback in the design process. The objective of this thesis is to investigate the roles of top-down and bottom-up processes, and how to integrate them in the robot design methodology.

The proposed approach utilizes several components from existing design methods. There are three main conditional loops within the proposed approach. The first loop focuses on defining the problem in a top-down manner through logical decomposition, defining technical requirements, researching solutions, and conducting a trade study. These four steps are done iteratively until reaching the bottom of the system, the most primitive components. This is followed by a modeling and analysis loop. This works from the bottom to the top of the design in preparation for manufacturing and validation. The final loop of the proposed approach focuses on validation and verification. The testing and manufacturing involved allows for alterations to the design to fulfill the original technical requirements. These three loops occur until a proof of concept is achieved. The proposed method is intended to be applied iteratively. The first pass of the method results in a proof of concept, while the second results in a preproduction prototype, and the third in a production model. This assembly of design elements provides a project flow that leaves little to be interpreted and is suitable for small design teams while still flexible enough to be applied to diverse robotics projects.

This thesis provides three case studies analyzing the application of the hybrid design approach mentioned above to robotic system development. The first study showcases a complicated system design with a small development team. The second case is of simpler construction with a smaller developer team. This simpler case better demonstrates the benefits of this hybrid approach in robotic system development due to the comparatively higher speed at which the system matures. The third case study shows how this same proposed approach can be applied to the design of a bottom-up controlled swarm. These case studies are for future designers to reference as examples of the hybrid design methodology in application, and what can happen when there is a lack of feedback in design. This proposed hybrid design method can encourage design practices in new engineers that translate better to industrial applications, and therefore encourage faster integration of new engineers into established design engineering practices.

Acknowledgements

This research was made possible by the NASA West Virginia Space Grant Consortium, Grant # 80NSSC20M0055, NASA Project # 80NSSC17M0053, and Alpha Foundation Grant # AFC820-69. I would also like to thank Yu Gu, Xi Yu, Jared Beard, Nicholas Ohi, and Trevor Smith for their help with this research.

Table of Contents

List of Figures

List of Abbreviations/Nomenclature

Accreditation Board for Engineering and Technology	ABET
Bottom-Up	BU
Computer Aided Design	CAD
Coordinated Universal Time	UTC
Degree of Freedom	DOF
Field and Aerial Robotics Laboratory	FARO
Global Navigation Satellite System	GNSS
Global Positioning System	GPS
Graphical User Interface	GUI
Interactive Robotics Laboratory	IRL
Internal Measurement Unit	IMU
Lunar Roving Vehicle	LRV
Mars Sample Return	MSR
NASA Procedural Requirements	NPR
National Air and Space Association	NASA
Top-Down	TD
West Virginia University	WVU
Verein Deutscher Ingenieure	VDI

1. Introduction

Every industry functions through the application of design methods. This essential operation is both a science and an art. There are generalized systematic approaches for engineers to follow. However, these processes still heavily rely on human creativity and experience to fill in the gaps and modify the methods on a per case basis. This human aspect of design is formed through practice, knowledge, and talent. These generalized methodological design approaches are necessary to establish a baseline for new engineers and are what mold the experience they accrue. These formal methods do well to provide direction, but must better reflect the practices and needs of industry in order to increase acceptance of these academic approaches. The evolution to these more practical academic methods will lead to faster integration of new engineers into existing structures and encourage more systematic practices in industry.

There are many attempts to formalize the "science" of design through methodologies. Most of these approaches can be boiled down to aspects from two major groups: top-down and bottom-up. Top-down describes how a system is governed by a group with a particular intent. These top-down methods are typically feed-forward in nature and focus on producing a result through the goal decomposition of the given task. For example, designing a means of robotically mowing a lawn can take many forms. However, this task can be broken down into simpler components, such as the cutting apparatus, drive train, power, computational, control, localization, perception, communications, and software subsystems. These subsystems can be broken down even further with this top-down manner to the most primitive level to define the system fully. Starting development at the bottom of these top-down primitive elements is where one method of bottom-up approaches can be introduced into design.

Bottom-up governed systems rely on many separate processes to determine the outcome. These methods are focused on iterative design and implementing feed-back from the interaction with the environment to determine subsystem composition. This results in the overall system taking form after these subsections are determined and the product essentially acting as an adapter between these subsystems. For example, evolution has demonstrated the ability for organisms to improve performance in their environments through mutation and natural selection [1]. In nature, these adaptations are accomplished without top-down knowledge or input, and success is determined by their behavior in the world.

This classification with two major classes encapsulates all formal design methods and a majority are some fusion of the two approaches. Commonly applied design methods utilize a core top-down approach with bottom-up elements, like feedback from experiments or verification according to stakeholder requirements. These formal methods have left it up to the user when to apply these checks. Some top-down approaches incorporate bottom-up elements into the core operation of their methodology. This is demonstrated through internal loops of synthesis and analysis, or periodic verification and validation of the system. Other methods demonstrate the ability to utilize mostly bottom-up construction methods with segments of top-down synthesis within the design levels.

A hybrid design between these top-down and bottom-up methods can be applied to nearly any system. An example is provided through a theoretical pipe inspection robot. This process starts with a top-down approach, by defining the task and breaking down that task into essential features. Then the bottom-up approach with top-down periods of synthesis can occur. The process specific components are defined and prototyped first, like the perception and drive train subsystems. Then the supporting components for those primary systems can be defined and implemented, like the computational, power, and communications subsystems. This process continues to go up through the levels of definition to complete the design. This example leads to the overall form of the system being determined by the subsystems, rather than the other way around. However, engineers' intuition is still needed to determine where to continue breaking down a system into subsections and appropriately design system requirements to solve a given task. This thesis strives to derive a practical and thorough hybrid top-down and bottom-up design approach from existing methods geared toward robotics applications that is easy for new engineers to interpret and utilize.

Through the development of this hybrid design method this work contributes:

- insight for the need and benefit of more application of bottom-up design concepts;
- interpolating a design method from commonly accepted approaches;
- applying this hybrid approach to three robot system design and development case studies;

The rest of this thesis is structured as follows. Chapter 2 provides a background of relevant topics. Chapter 3 interpolates and discusses when to apply the proposed design method. Chapters 4, 5, and 6 are the case studies of this design method in practice. The rover in Chapter 4 applies the hybrid design method to a complicated system construct with a small team of engineers. The rover in Chapter 5 applies this proposed method to a simpler system and demonstrates application over significant system maturity. The third robot in Chapter 6 demonstrates this design method applied to the software development side of robotics. Finally, Chapter 7 wraps up with concluding remarks.

2. Background

This background has been broken down into four sections. First, the Bottom-Up and Top-Down section discusses the definition of these classifications and provides examples of these types of systems. Second, the Design Methodologies section explores commonly accepted formal design approaches and their prominent features. Third, the Decentralized and Centralized section defines and clarifies the distinction between these two types of systems in robotics. Fourth, the Consensus section discusses the principles of agreement protocols and their significance to robotics.

2.1. Bottom-Up and Top-Down

The terms top-down (TD) and bottom-up (BU) are often used to describe how a system is governed. These terminologies can apply to psychology, economics, engineering, management, politics, and many other disciplines. [3,4,5,6,7,8] In a general sense, TD can be denoted as something built, or controlled to achieve a specific goal determined by an individual or group. In engineering, TD methods are the traditional development approaches in industry. [9] Generally, the designer is given an objective, decomposes the problem into solvable goals, explores solutions, models the proposed solution, and synthesizes a system which is focused on achieving that original objective. This TD system only takes into account the situation as well as the designers' account for the scenario. Additionally these systems may not do well in the presence of significant changes to the application [2]. For example, a logistical robot from a warehouse would struggle to operate in an unstructured environment without modification, or a robotic manipulator from an automotive assembly line would struggle to effectively pick apples without modification. In economics, TD can describe how a model is constructed from high level predictions [3,4,5] or even how policies are implemented as a blanket effect [6]. In psychology, TD can refer to how humans anticipate interactions with the world due to prior experience, rather than simply reacting[7]. In general, purely TD methods focus on feeding forward a detailed control or design to fulfill the objective and do not provide room for feedback from experiments, or modularity in application.

In contrast, bottom-up (BU) typically refers to something that is determined in a more decentralized manner. As many separate processes determine the outcome, or design, of a system [8]. For example, a forest can be likened to a BU system in the sense that every plant, rock, and creature is not controlled or planned by any one group or individual. Instead their behavior is governed by their inherent biological and physical rules in response to external stimuli. BU approaches in engineering utilize the environment, or scenario, to determine the design of the modules to facilitate the overall system objectives. This feedback into the design is typically witnessed over several iterations [2]. Although the system may not operate well outside its intended application, the iterative design process and use of modules result in an adaptable design. In economics, BU can describe how a model is constructed based on gathered information [3,4] or even how policies are implemented on a per case basis [6]. In psychology, BU can refer to how humans react to the world through senses, rather than anticipating results

[7]. In general, purely BU methods focus on feedback into the policy, control, or design based on performance. Although this approach is much more practical in producing end results, it is more convoluted to synthesize an initial form to unify the subsections with a purely BU method.

These polar opposite ideas of TD and BU are still very codependent on each other. There is no purely TD or BU system [2,8]. Every effective design method in industry utilizes TD and BU to some degree [10,11]. For example, the Spiral design model, discussed later in more detail, operates in a cycle of TD design and development with phases of BU feedback and validation over an iterative process. Even though this Spiral method would be regarded as a TD method it has BU elements present. The iterative prototyping step of generic TD processes is a core aspect of how BU is applied to design and BU designs still require some sort of TD synthesis after the analysis step [2]. Ultimately, BU designed systems are modeled by humans, introducing TD elements, but the compliance and modularity added to the system promote robust performance and potentially lead to new features, or behaviors, emerging during the BU design cycle.

2.2. Design Methodology

Design can be regarded as an invisible study in society. Many take this research topic for granted due to its commonplace application in everyday life. Coordinated Universal Time (UTC), Global Navigation Satellite System (GNSS), or electrical grid management are considered invisible utilities to the general public. Most individuals learn their own design methodology through experience in industry and their personal life. Personal design methods vary greatly between professions. However, design in the engineering sense will be the focus in this section.

The Accreditation Board for Engineering and Technology (ABET) defines Engineering Design as, "... identifying opportunities, developing requirements, performing analysis and synthesis, generating multiple solutions, evaluating solutions against requirements, considering risks, and making trade-offs, for the purpose of obtaining a high-quality solution under the given circumstances" [10]. All of these steps are top-down except for the "evaluating solutions against requirements" steps. The user takes in feedback from the design's characteristics at this stage, making it bottom-up. These common ideas of design are a great baseline, but this is still an abstract concept. There has been ample criticism in the design community of the state of design methodologies. It is a point of concern that students are not made aware that these methods should be taken with a grain of salt, or why the method is structured as such [12]. Many design methods in academia often omit how to go about the major steps of the process, or why, let alone take into account resource limitations seen in industry that completely alter steps found in the ideal, academic design process.

Despite this criticism, Howard et. al's summarization of the diverse design methods discuss that there are some common properties of the processes between vocations. A generalized process model consists of: "*establishing a need, analysis of task, conceptual design phase, embodiment design phase, detailed design phase, and implementation phase*" [12,13]. These generalized stages of activities cover a wide range of design applications. It is generally

accepted that these methods must happen iteratively to produce good results. However, this non-specific design process will not apply to every situation or profession. It is commonly regarded that the ability to modify these general processes is one of the most important skills of designers [14,15]. In fact, most applied engineering design processes are some variation of the "Verein Deutscher Ingenieure" (VDI), Association of German Engineers, 2221 design process shown in Figure 2.1 below, the Spiral Design Model by Boehm, or the V-Model by Forsberg and Mooz [12,15], shown in Fig. 2.2.

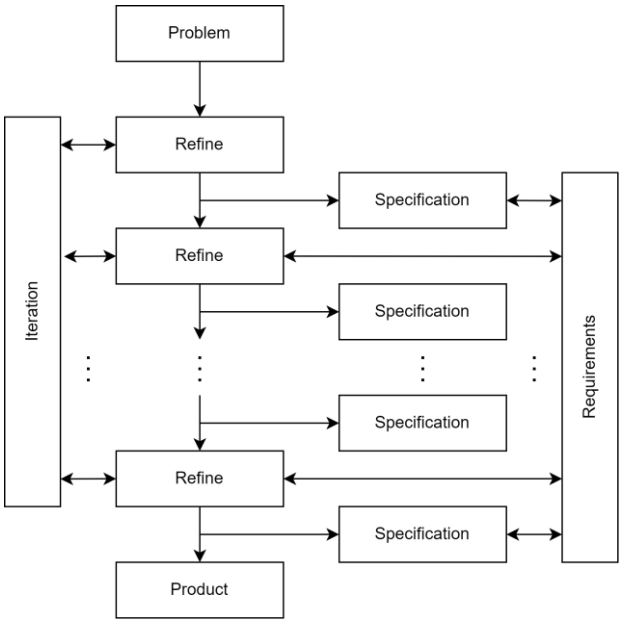

Figure 2.1: Diagram Reproduced by Author Based on Common Design Process from Association of German Engineers, VDI 2221 [16]

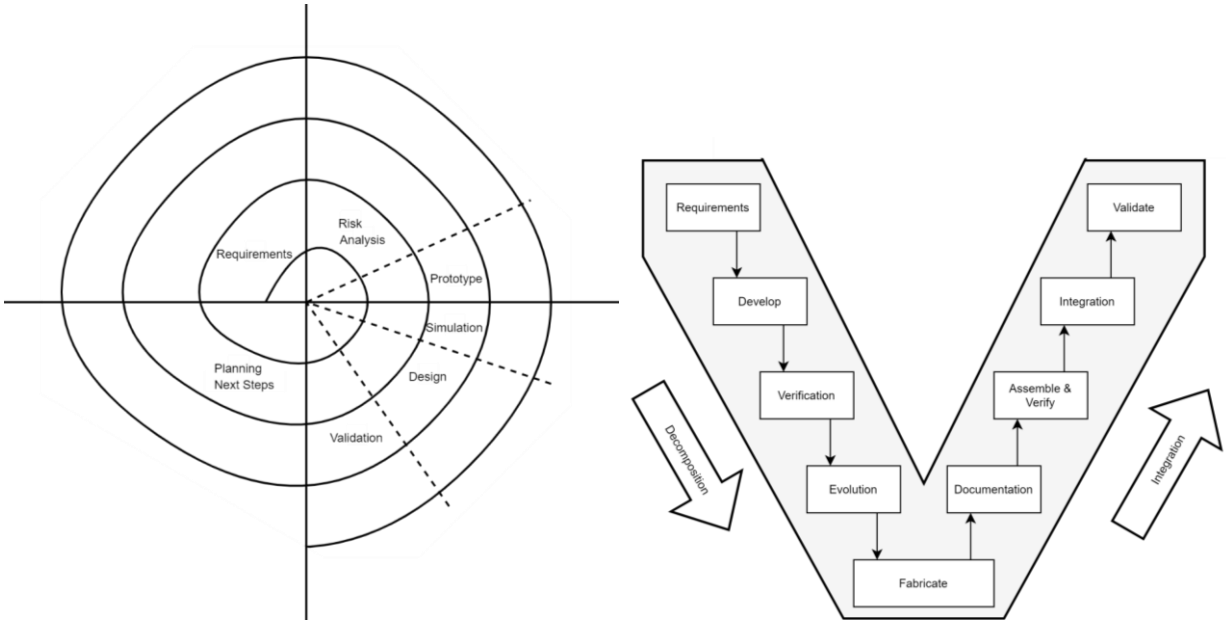

Figure 2.2: Diagram Reproduced by Author Based on Spiral Model of Software Development [17] (Left) and V Model of Software Development [18] (Right)

One of the main distinctions of design processes between professions is how much emphasis is placed on a particular step [19]. This allows other disciplines to better provide solutions for their industry's needs. Gericke and Blessing stated, "*Civil engineering may provide approaches to deal with the separation of development and production. Software and knowledge based engineering may provide approaches to include user issues more explicitly. The abstract, function-oriented approach in software and electrical engineering may provide solutions to deal with mechanical systems at a more abstract, functional level*" [12].

In multi-disciplinary designs, like those found in mechatronics, the VDI 2221, Spiral, and V-Model methods do not encompass a practical process that applies to the entire project. As a result, VDI 2206, shown in Figure 2.3 below, was developed specifically for robotic applications. This method utilizes elements from the VDI 2221, Spiral model, and V-model processes.

The macro level model is a flow to follow in the overall project and can help establish milestones for management to track progress. This is based on the traditional V-model, but emphasizes the need for parallel processes, flexibility of the design process, and constant verification of the design. The micro level model is geared towards the individual designer and operates within the system design, system integration, and domain-specific design sections of the macro level process. This begins with a parallel process that takes into account both client and designer defined goals. The synthesis/analysis loop allows for the exploration of multiple solutions within the same iteration. This micro level process emphasizes the need for repeating the process if the results are not satisfactory, and investigating alternative designs. The micro level process is where the VDI 2206 process gets inspiration from the VDI 2221 process.

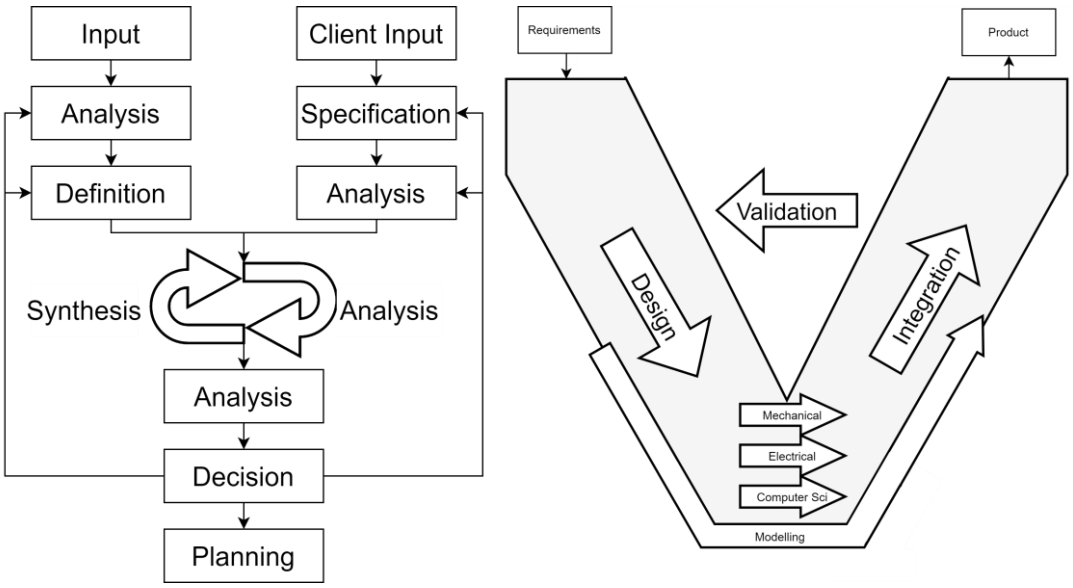

Figure 2.3: Diagram Reproduced by Author Based on VDI 2206 Micro Level (Left) and VDI 2206 Macro Level (Right) [20]

Feedback from validation at the macro level helps to shape future iterations of the VDI 2206 process. This intuitive feedback within the micro level should help to reduce the number of macro iterations. However, completing this macro level process does not necessarily result in a finished product. This process is intended to be implemented repetitively to increase system maturity and reach significant milestones, shown in Figure 2.4 below. This is where the VDI 2206 process pulls in elements from the Spiral Model used in software development. For example, the first pass of this process could result in a concept model, the second pass could result in a functional model, and the third iteration could result in a first physical prototype.

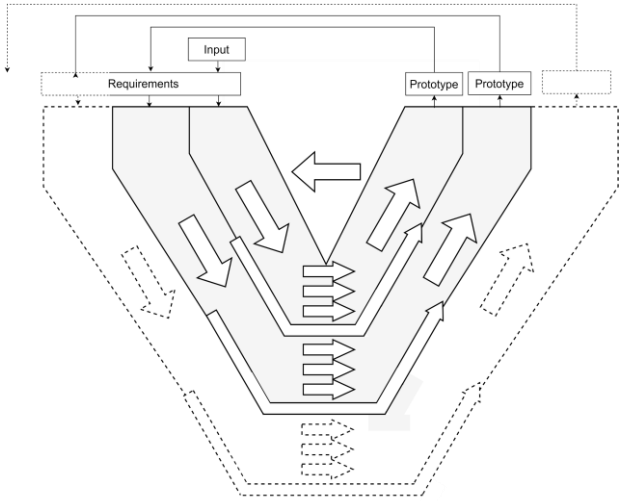

Figure 2.4: Diagram Reproduced by Author Based on VDI 2206 Product Maturity Cycle [20]

Another commonly accepted design method is that of the NASA Procedural Requirements (NPR) 7123.1 method, shown in Figure 2.5. This outlines the method of Systems Engineering with mechatronic systems used in their Systems Engineering Handbook [21,22]. This NPR 7123.1 Systems Engineering method is broken down into three main processes: system design, product realization, and technical requirements. These three overarching processes are applied recursively to every level of decomposition of the system until reaching the most primitive elements.

The system design portion focuses on establishing the task. This involves defining the stakeholders' expectations of the product, then defining the corresponding requirements for those expectations, further decomposing those requirements into manageable problems, and selecting potential solution for exploration.

The product realization section focuses on ensuring cohesive integration with adjacent components. Firstly this is done by implementing the realized end products from lower levels. Then, the primitive level solution is integrated with the overall system structure. This subsystem solution is then verified if it truly meets the criteria from earlier steps and meets stakeholders' expectations. These are all done before transitioning to the next layer above the current component.

The technical requirements section acts as a tool for project management. These steps emphasize taking the time to plan future steps, checking overall project progress, assessing potential risks, clear communication between team members, and documenting results. All of these management steps assist in analyzing the progress and how to improve further.

These seventeen steps do not all happen before proceeding down to the next level, otherwise called Work Breakdown Structures (WBS). A product design can be regarded as bottom-up if these lowest levels are defined first and work up through the WBSs from there. In contrast, a design can be regarded as top-down if the higher level WBSs are formulated first with the lower level components determined in subsequent steps.

Following the flowchart in the figure below, the system design steps for all subsystems happen before working back up the chain through the product realization steps. All of these steps happen in parallel with the technical requirement steps, as project management is a constant endeavor.

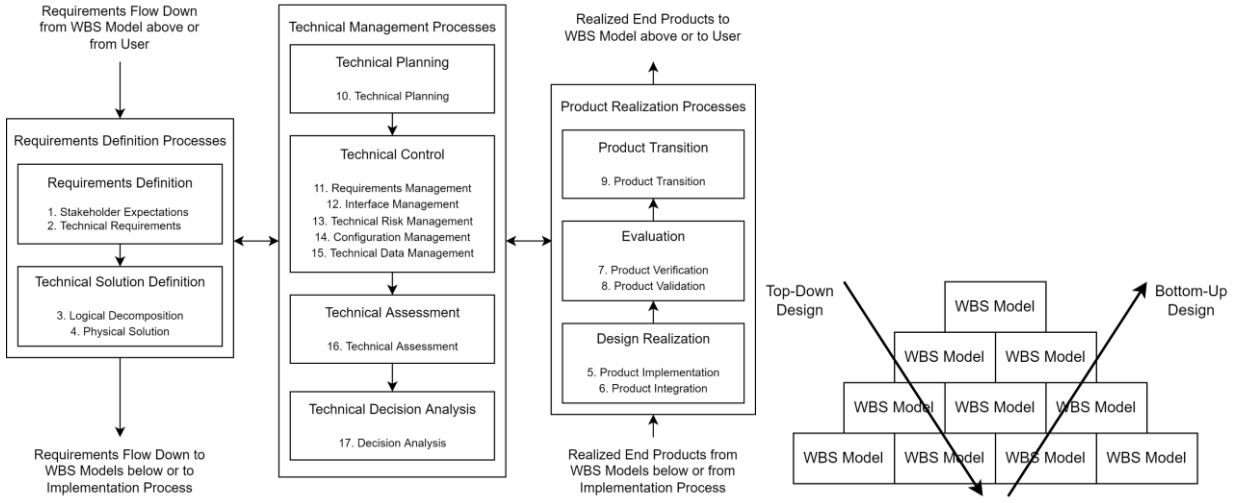

Figure 2.5: Diagram Reproduced by Author Based on NPR 7123.1 Systems Engineering Method [21,23]

This NPR 7123.1 Systems Engineering Model works within the outlined NASA Project Life Cycle in Figure 2.5. The system design processes are represented as section 4.x. The need to implement these four steps recursively until reaching the lowest level of the project is clarified in Figure 2.5. Only then can the five product realization processes start progressing back up through the subsystems. This figure also emphasizes how the technical management is constantly in parallel to these nine steps. This documentation and formalization of the design process are increasingly necessary at each major design review.

These iterations of the NPR method occur during each major design stage of the project. There are considered to be seven major stages to NASA's projects. This begins with the "concept studies" to define the feasibility, challenges, cost, and other requirements to put together a first draft of the project. The project really begins with the "concept and technology development" stage determining the basic concepts, plans, and requirements to fulfill the project. The

“preliminary design and technology completion” stage aims to produce a barebones prototype to meet mission needs. The “final design and fabrication” phase occurs midway through the project life cycle. This stage focuses on actually producing the final product hardware and developing the initial software packages. The “system assembly, integration, test, and launch” phase brings the entire project together and ultimately producing the final product. The “Operation and Sustainment” stage executes the desired mission. Finally, NASA projects wrap up with a decommissioning procedure to close out the operation and compile data for analysis.

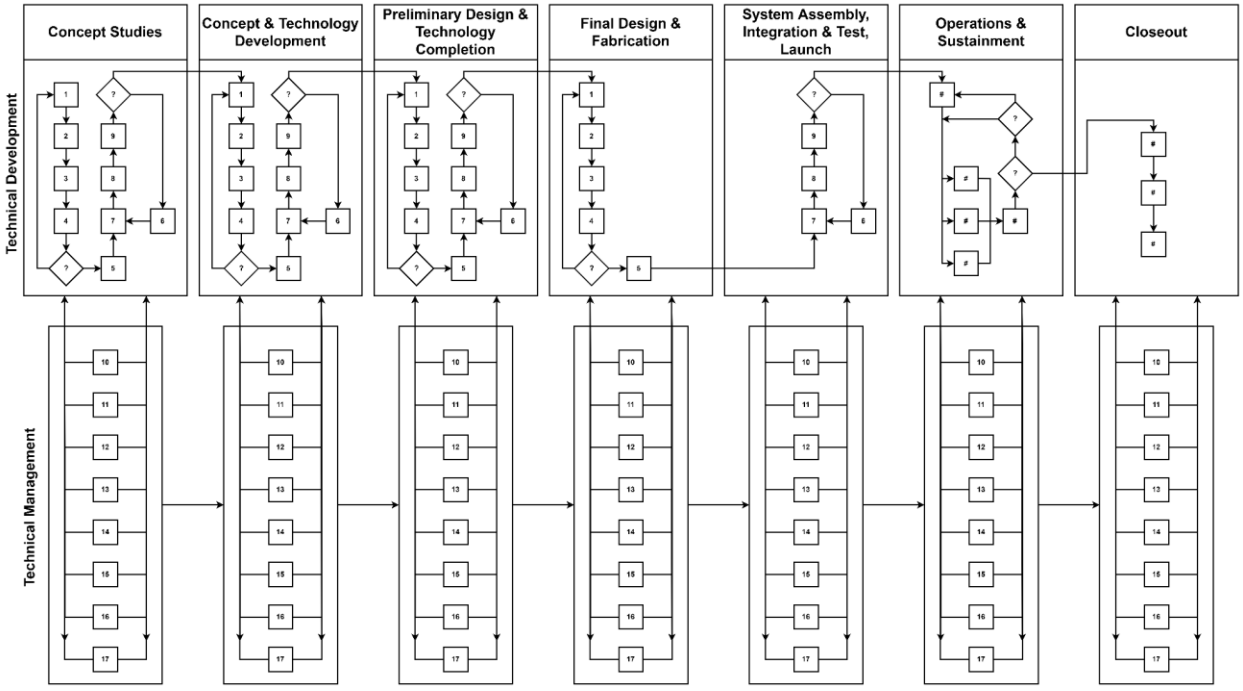

Figure 2.6: Diagram Reproduced by Author Based on NASA Project Life Cycle [21]

The five major methods discussed thus far do not encompass all processes used in industry, but do cover the commonly applied design methodologies in practice. All of them share elements of bottom-up and top-down methods to varying degrees. VDI 2221 is a busy flow chart, but it consists of mostly parallel top-down processes. There is bottom-up feedback to the system design and system requirements, but it is not well defined when to apply these steps.

The Spiral Model is much more detailed and practical, but is a very single threaded implementation of the design process. This also consists of mostly top-down steps with periods of feedback, a bottom-up feature, located on the lower vertical axis.

The Forsberg-Mooz V model is also a very serial process that consists of mostly top-down processes. The only feedback in the system is the general verification statement, and does not illustrate the need to update the system requirements. This bottom-up feature is haphazardly applied, leaving the designer to decide when feedback is necessary.

The VDI 2206 model uses both top-down and bottom-up approaches throughout the process. The macro level emphasizes a constant feedback loop in the design process. This makes

the macro level top-down just as much as it is a bottom-up process. The micro level utilizes a top-down approach within a feedback loop. Although this is a bottom-up element, the micro level is still a mostly top-down component. Many robotic designs utilize off the shelf components with simplified interfaces between these systems. These subsystem parts drive the VDI 2206 process, resulting in a dominantly bottom-up design process. The form and function of the system is still a top-down process with periods of bottom-up feedback. This design method results in the robotic project becoming a glorified adapter for the modules that fulfill the mission objectives.

The NASA System Engineer methodology emphasizes the top-down development of a system, and implies that bottom-up feedback is constant. This approach is thorough and effective at managing massive engineering development teams. However, this is not necessarily practical for smaller companies or design teams. Applying this method to smaller groups can lead to increased burden on team members to formalize steps they may intuitively do in the process.

These five major approaches demonstrate that design methods are predominately top-down with bottom-up applied as an afterthought, or as a delayed form of verification. These methods also have various aspects that are either not appropriate for small team applications, leave too many details open to interpretation, or fail to incorporate essential aspects of the learned design methods. These self-learned methods strive to balance research, synthesis, and feedback while maintaining monetary, effort, and time constraints. Some of these intuitive constraints are often omitted from design methods taught in academia, and ultimately lead to industry reteaching design practices to new hires.

Engineers are taught how to research and synthesize solutions via task decomposition in academia. These top-down methods are often instructed without reference to formal design methods, but typically resemble the synthesis focused VDI 2221 process. It is essential to understand this method as an engineer and cutting edge developments are made through synthesis. However, bad designs can originate from synthesis as well and this mindset focused on synthesis can often plague the development of products in industry. The lack of practicality and how to implement the steps outlined in this ideal method often leads to rejection by industry [15,24,25,26]. This results in employers often training new hires how to practically navigate product development. The higher standards of quality, need for economically and technologically competitive products, and desire for rapid progress force a different mindset onto these new engineers. However, there is often less opportunity to explore different prototypes in industry despite these requirements. This is mostly due to limited resources and the desire of experienced engineers to stick with the techniques already implemented. This restriction on prototype exploration and focus on top-down synthesis often leads to users continuing this habit for the entire project. This results in first prototypes heavily influencing the final product instead of experimenting with more diverse designs.

Although it is good to pull ideas from experience, there are great benefits to veering from the status quo and re-evaluating design choices. These changes can seem minor at first glance, but may completely alter the quality and opinion of a product. For example, the original Boston

Dynamics Spot platform, now called Spot Classic, evolved from its hydraulic actuated origins to its fully electric system in the later 2010's [27,28]. This advancement not only led to a simpler design, but the quiet system also improved public perception of Spot. This advantage of producing alternative designs to provide feedback is intuitive. However, these expensive full prototype explorations are usually undesirable in industry and it is often unclear when to do so in several formal design methods. Applying more feedback, or other bottom-up methods, in industry can provide pathways to improve products.

There is a lack of purely bottom-up design methods. This academic pursuit has been realized through the use of evolutionary and reinforcement learning based automated design methods to better imitate processes witnessed in nature. For example, an increasingly common method of exploring optimal solutions in simulation is through a genetic algorithm. In every generation, or iteration, agents attempt to complete the task. These agents all have properties resulting from mutation, selection, or crossbreeding of the dominant agents from the previous generation. This can be applied to a wide array of tasks that need optimizing. These projects can include: structural design [29], neural networks [30], printed circuit board design [31], and much more. These software generated designs are a good baseline for improving design practices, but it is difficult to completely automate the entire process. The need for co-designing the often conflicting interests of mechanical, electrical, and software systems in parallel tremendously complicates the design process. Even minor modifications to the mechanical design can have significant impacts on the electrical needs and software capabilities of the project. This is true for any of the involved disciplines of a mechatronic system. Utilizing general purpose hardware and making up the difference with software only goes so far. Designing complicated systems is still an art due to the need for adapting these formalized design processes to a particular project and knowing where to break down or probe the product further.

This thesis explores a method interpolated from the NPR 7123.1, Spiral Model, and VDI 2206 approaches to provide a straightforward method geared towards encouraging good design practices in new engineers. The NPR 7123.1 approach is thorough in nature and well defined. This begets an overall strong design methodology, but is not friendly to new students due to its definition of planning, management, analysis, and assessment occurring in parallel to the active design steps. The Spiral Model encompasses an excellent pathway from problem statement to product realization, but its open ended descriptions may prove confusing to students who have not done much design work before. The VDI 2206 model's structure does cover a realistic design method, but also leaves feedback and subsections open to interpretation. A hybrid of these approaches would strive to be well defined and thorough like the NPR 7123.1 method, while being approachable and clear like the Spiral method, and resemble the structure and feedback loops of the VDI 2206 method.

2.3. Decentralized and Centralized

In a general sense, the terms centralized and decentralized describe how a system is structured. A system is centralized when all actions are controlled by one member, or controlled by a distributed system with global knowledge. For example in robotic manipulators in a factory, each robot has a centralized controller that is aware of the robots state and collects information from sensors to decide what action to take next. This top-down knowledge of the system's status is by far the biggest advantage of centralized distributed systems. This global knowledge makes effective coordination of multi-robot systems much easier to achieve. However, this structure requires good connection between the agents in the distributed system. For example, sensors distributed across a factory can collect data for a centralized digital double. This connection requirement isn't always feasible when operating in unstructured environments, or with mobile systems. These centralized systems do not scale well to large groups of agents due to communications and computation bottlenecks and may not be robust to failures.

In contrast, a system is decentralized when it is composed of many units that make their own decisions based on local information. These agents do not necessarily have direct knowledge of the rest of the system. This decrease in the information communicated simplifies the processing each agent needs to do. For example, a swarm of birds flying in a flock while they can only observe immediate neighboring birds is an example of a decentralized system operating in a bottom-up control manner. These properties do well to scale with large systems, but come with their own challenges. These systems rely heavily on constant communication to provide agents with information for local interactions and localization is more difficult to achieve. These challenges are often worth the additional benefits that decentralized systems bring. These systems are typically more tolerant to failure of some units in the swarm. However, it is more difficult to accurately predict and coordinate the system as a whole due to this lack of global information.

There are many systems that apply both centralized and decentralized structures. In electricity generation, power plants are part of the centralized power grid system. Residential solar panels contribute power to this same system without knowledge of the rest of the infrastructure, and are currently decentralized contributors [32]. The advent of new "smart" electric panels aims to change this decentralized nature of photovoltaic and other residential power systems [33,34]. In computer science, decentralized websites are mostly hosted in major data centers, or centralized facilities, and distributed via centralized internet service providers. However, the introduction of blockchain has brought about the idea of decentralized internet server facilities for the decentralized website services, otherwise called Web 3.0, to be distributed through these centralized internet service providers [35,36] .

As seen in these examples, the opposing ideas of centralized and decentralized are still very codependent on each other. As there are no purely decentralized systems. For example, a drone swarm that utilizes decentralized computation, communication, and control would be declared a decentralized system. However, in most cooperative applications there still is a centralized knowledge that the other agents in the system are bound by the same rules. In many

decentralized systems, agents that follow different programs or physical rules will not be able to cooperate and can cause the whole system to not perform well [37,38].

2.4. Consensus

A means of agreement between units is needed to encourage cooperation within multi-agent systems, referred to as consensus. The term consensus has applications in many fields, such as physics, computer science, robotics, politics, and much more. [39,40,41,42,43,44,45,46,47,48] The term consensus refers to the collective decision making of multiple agents in a system to agree on an outcome. In computer science, multiple users connected to a game server require a means of synchronizing the game world continuously for all users. This consensus of the game world is accomplished through continuous clock synchronization, state estimation, state prediction, and state reconciliation [39,40]. In politics, consensus is sometimes found, but not always achieved, in the process of formulating and passing legislation [41]. In multi-agent robotics, this can take on the form of rendezvous, formation stabilization, and formation movement (flocking) problems [42].

There are two major schools of consensus in these multi-agent robotic systems: centralized and decentralized consensus. Centralized consensus is essentially an extension of the same approaches to coordinated control seen in single agent systems. The agents within the centralized multi-agent system are extensions of the central decision maker. Decentralized consensus methods are the more challenging and valuable approach for multi-agent systems. These decentralized systems are typically more robust to faults, able to scale to large systems more easily, and able to perform more complicated tasks with less system complexity as compared to centralized systems. Decentralized multi-agent system consensus has taken significant inspiration from equivalent scenarios in nature. These include the flocking of birds, ants working together, wolf packs hunting, and many other similar scenarios.

There are several classifications of collective decision making in robotics. Leader-follower consensus systems consist of agents listening to a selected leader. For example, a leader in a group of mobile robots could follow a high level path given to it while the followers in the group simply keep formation around the lead unit [43]. Variations of this task can dynamically select leaders of the swarm. This voting system is a core aspect of group consensus, another major classification of consensus tasks. These systems exchange information to determine an agreed truth. For example, distributed sensors measuring temperature need to agree on a correct value through consensus protocols [44]. Multi-consensus systems tend to resemble group consensus problems, but add the ability to track multiple truths in their protocol [45]. For example, a swarm of robots could observe multiple objects of interest traveling through their field of view. These robots can work together to keep track of the target while it is out of view of another robot [46]. These broad classifications of consensus in robotics utilizes agreement protocols to facilitate deciding on one output in the swarm.

Some of these methods of group consensus are governed by simple mathematical functions that always converge. In group consensus, reaching convergence by difference and

differential equations are excellent examples of these primitive, but powerful equations that can reach agreement through a decentralized multi-agent system [42]. Average consensus algorithms are also quite simple and demonstrate the ability to accommodate many inputs [42,47]. Best-of-n decision making utilize increasingly complex protocols, but enable more intelligent decisions to be made by the decentralized swarm [48]. There are many methods of reaching consensus in computer science, but these approaches and classifications of consensus problems are of particular interest in robotics.

3. Technical Approach

The proposed hybrid robot design method will be composed of several loops like the NPR 7123.1 method, but the internal steps of these loops better resemble the Spiral method and there are no parallel steps to the loops, as seen in the NPR 7123.1 approach. These are done to simplify the instruction of this method while remaining adaptable to different robotic projects.

This hybrid of several formal methods aims to better align with the needs of flexible design in industry while remaining well defined enough for upcoming engineers to sufficiently understand and build up practical experience. To promote good design practices a method should balance research, synthesis, and feedback while respecting monetary, effort, and time constraints. The academic design methods resembling the VDI 2221 process typically neglect the feedback, a bottom-up feature, of these six golden traits. Inexperienced engineers still receive feedback from instructors, although that is usually after the end of a project. The application of validation, verification, and experimentation in the design process can drastically improve overall learning, even if less material is covered.

Not all of the steps outlined in the previously mentioned methods in this thesis are appropriate for robotics projects. The VDI 2221 method is simple, and a good introduction to design as a whole. However, this method fails to mention how far to take each step in the process, or what should the end product be of that iteration of the process. This obscure description leaves the VDI 2221 method open to interpretation and results in significant variations in the application of this design method. The Forsberg-Mooz V model does resemble the structure of other accepted techniques, like the NPR 7123.1 process. Unfortunately, this less detailed process implies that it is only to be applied for one iteration, and focuses on progressing through the synthesis of the product. This lack of feedback in the methodology severely hampers the ability of the process. The NPR 7123.1 and NASA Project Life Cycle processes are very detailed. NASA has developed and proven these design methods over years of practice. These methods are thorough, but may be too much to manage for new engineers and small teams of 4 or less. The VDI 2221, VDI 2206, Forsberg-Mooz V Model, and NPR 7123.1 all imply that feedback is constant. This unfortunately makes it unclear to less experienced designers when is the best time to apply feedback into the system design. The VDI 2206 and Spiral approaches at least make the effort of when to introduce feedback into the design process. The Spiral method is much more detailed and appears very approachable for students, but only discusses application to software. The VDI 2206 process has feedback cycles built into its core processes, but needs to further clarify when to stop a particular cycle. All of these methods have some aspect of them that make them more difficult for inexperienced engineers to digest, or not appropriate for applications to robotics. Many of these issues can stem from too much detail, not enough clarity in the definitions, obscure design flow, and where to integrate bottom-up methods.

There are still many useable components of these methods. The VDI 2206 micro level and Spiral methods are excellent examples of all six traits and integrate feedback as a core step in these methods, but still leave the many details and decisions of those steps up to the user. In contrast, the NPR 7123.1 method provides incredible amounts of detail, but implies that

feedback is a constant parallel process and ultimately still leaves it up to the user when to apply feedback in a small team setting. An interpolation of these three methods strives to provide detail, examples, and feedback loops like the NPR method while drawing steps and structure from the VDI 2206 and Spiral processes. These assets are compiled while keeping the hybrid approach defined in a way that reduces the need for interpretation and still flexible to diverse applications.

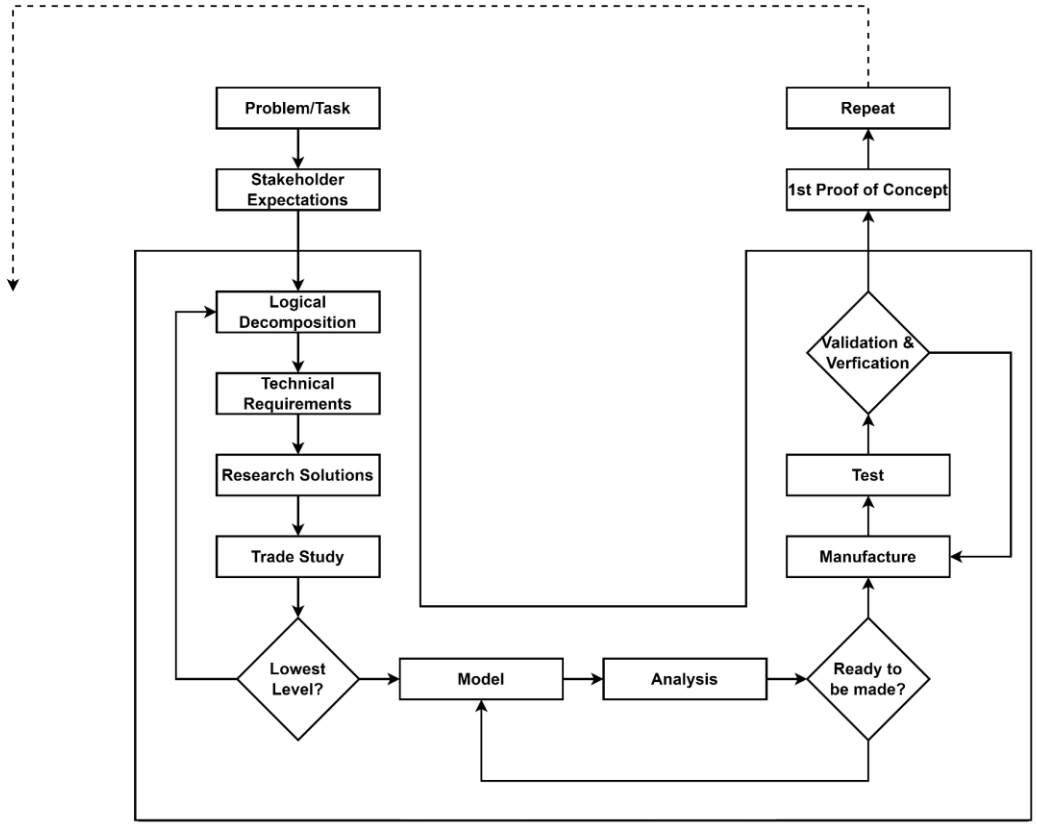

Figure 3.1: Proposed Hybrid Design Methodology

This proposed design method shown in Figure 3.1 begins a top-down focused process that iterates problem decomposition, requirement definition, solution synthesis, and trade studies through a loop. Logical decomposition is essential for any project, and makes for a collection of simpler problems to solve. That decomposition drives the technical requirements of this level in the design. These levels are the hierarchical breakdown of the system determined through decomposition. For example, this can start at the system as a whole, then proceed to the subsystems, and further to the components needed to make those subsystems function. These systems are continually broken down into more primitive components, like bolts, resistors, and other off the shelf products. The technical requirements of the system should be kept as broad as possible as to avoid defining a solution in the goals. Then the team conducts research of potential solutions, and considers all solutions, no matter how infeasible. The purpose of the trade studies is to explore the pros and cons of the systems, and reduce the role of human preferences in

decisions. This objective thought process should bring out the best fit solution of those found at the time, and justify why one or multiple options are considered for further analysis. This is done repeatedly until breaking down the system into the most primitive components. The steps up to this point should only be defining general features and properties of the system. Just like the technical requirements, the goal of the first loop is to define the system as broadly as possible. The technical requirements of the lower levels of the system will fill in the details through the following modeling loop.

Only after these primitive components are defined can modeling and analysis begin. This modeling process starts at the lowest level and works bottom-up to define the overall design. This bottom-up approach allows the necessary equipment to dictate the structure and form of the system. Although this seems intuitive, it is not uncommon for a system's structure and form to be designed first and then attempting to find components to fit that system later. If this risky decision is used the system design is usually altered significantly, and this should not be done when designing and manufacturing in parallel. When designing these systems, it is important to keep modularity, manufacturability, assembly, maintenance, tolerance, and cable management in mind. These features will greatly increase the ability to adapt and improve the system as development and integration continues in later steps. Progressing onto the review phase only occurs once the model reaches a state of defining the overall system. The reviews are intended to verify the product design against the technical requirements defined earlier in the process. This can include finite element analysis, dynamic motion studies, thermal and vibration analyses, economic feasibility studies, sensor placement studies, scale model construction, simulating operation, manufacturing studies, and many other methods of validation. This cycle of reviewing and modeling occurs repetitively to provide feedback into the design from all the mechanical, electrical, computer science, various engineers and technicians involved in the project. This multidisciplinary review gradually defines the system from a geometric representation to a fully manufacturable CAD model of the product while balancing the needs of all the subsystems.

The final assessment of the readiness of this system must be thorough before progressing onto full system manufacturing. It is best to include the other trades involved in the manufacturing and implementation of this system in the reviews to produce more feasible and realistic designs. There may be some need for improvising unforeseen issues on these initial prototypes, but the previous reviews should catch a majority of these potential issues. These prototypes are where a majority of learning and validation occurs. There may be some phenomena that simulations and analyses did not catch. These studies can only capture so much detail, and are limited based on any information omitted from their models. There may be a need for subtle alterations to the components or their manufacturing processes to increase quality or ease of assembly.

These approaches handle individual levels of design, but a method of overall managing project flow is also needed to successfully apply the aforementioned process on a project-wide scale. NASA’s Project Life Cycle, Spiral, and VDI 2206 macro level all have reasonably defined project stages. Although the NASA proven method is thorough, it is often not practical to

implement on smaller scale projects, or with new engineers. The VDI 2206 macro level and Spiral methods provide an easy to understand course of action, but simply need more detailed descriptions and examples.

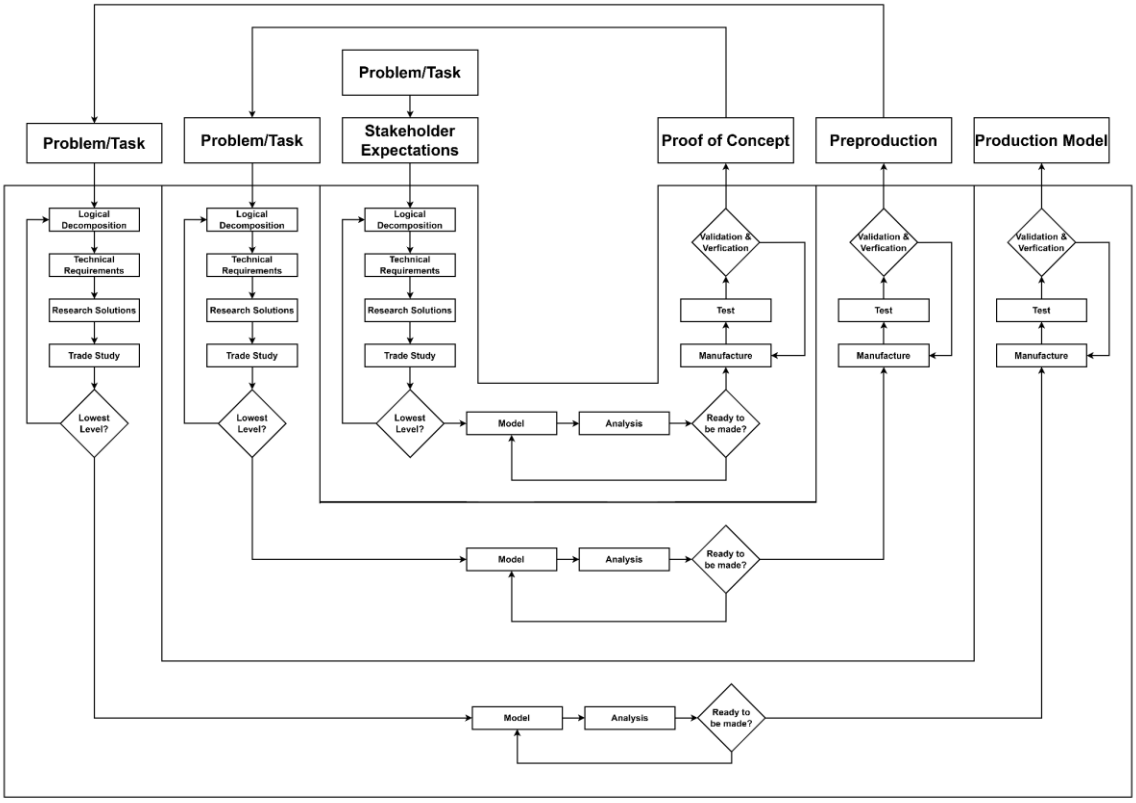

Figure 3.2: Proposed Hybrid Design Approach Macro Level

These three main loops shown in Figure 3.2 are inspired by the previously mentioned design methods. These identical loops are done over several iterations to achieve progressive milestones of the project like the VDI 2206 macro level process. These are not limited to any particular set of milestones. For example, the first iteration can start with a proof of concept prototype, the second pass could produce a preproduction prototype, and the third iteration could result in a final product that is ready for mass production. The number of prototypes within these loops really depends on the application of the system, level of quality needed, and the alternative subsystems considered. It may be beneficial to develop multiple proof of concepts for a particular subsystem, which benefit from the aforementioned modularity. This hybrid system design method is demonstrated through the case studies below.

As a whole, design in robotics does benefit greatly from experience, but the method outlined in this section presents a means of producing results in small teams through cycles of synthesis and feedback while reducing the ambiguity of the order of operations. This method should still be approachable for student engineers, and transferable to application in industry.

4. Case Study #1: Complicated Robot System Design

4.1. Introduction

The West Virginia University (WVU) Interactive Robotics Laboratory (IRL) has developed a planetary rover research platform called “Fast Traverse”. This four wheeled robot is equipped with independent actuated suspension and steering mechanisms. Fast Traverse is a test bed for path planning autonomy and how scientific instruments can assist a rover in determining safe paths.

The development team has included many members over the years. Yu Gu, Scott Harper, Nick Ohi, Conner Castle, Dylan Reynolds, Jared Beard, Benjamin Buzzo, Dylan Covell, Jonas Bredu, Chris Brindle, Eric Swanson, Gabrielle Hedrick, and Spencer Regnier have all contributed to the development of this rover and their specific works are emphasized in the design process below.

4.2. Design Process

This robot starts off with a significant challenge in the proposed design process. The very broad system constraints and considerations make it difficult to define the task to be fulfilled. Fast Traverse is to be a general purpose rover research platform aligned with NASA’s future planetary exploration concepts, like the Mars Sample Return mission [49], and accommodate other unforeseeable research needs that are similar in nature. These two vague requirements mean that the rover platform needs to be capable of accommodating a wide range of behaviors for algorithm testing and highly modular in nature for future mission configurations. Through deductive reasoning, this means that operation time between charges, weight, and complexity are of lower concern in the design considerations as compared to a robot designed for a specific application.

These few requirements gave a lot of flexibility to system requirement conception and desired features of the robot. The drivetrain would have to loosely resemble the capabilities of a NASA rover, but any solution could be chosen that facilitates a wide range of behaviors and accommodates NASA’s research objectives. The ability for the rover to cover ground at higher speeds and to push path planning algorithms further than its Martian counterparts fulfilled the need for this system to align with NASA’s exploration concepts.

To better align with these exploration concepts, NASA’s own work served as valuable input while researching potential solutions. Many rovers deployed by NASA demonstrate a few common traits, but the first dominant drivetrain trait that comes to mind is the modified rocker bogie suspension. This passive suspension system is very capable of conforming to terrain features, but increases system weight, cost, complexity and is only applicable to slow vehicles. All Mars rovers have some form of steering; for example, Curiosity and Perseverance rovers apply independent steering to the rocker-bogie suspension to provide agile movement and directional control at various speeds. Through observation of various planetary rover systems, it became clear that this independent steering system was a desired core feature of the robot. It was

less clear from the analyses on whether the commonly observed rocker-bogie system was adequate for the project's needs.

Seeing that the system is intended at traveling faster than the current Mars rovers, what speed goal would the system aim for? At the time of designing this project, the record distance traveled in a single Sol was achieved by Curiosity was 143 m [50]. (That record has since increased to 320 m with Perseverance [51].) Future missions, like the Mars Sample Return (MSR) mission, have strict time constraints on the large distance to cover. MSR only has 687 Earth days, or one Martian year, to bring samples back to the launch vehicle [52]. MSR may have to cover up to a 10 km distance[53] before reaching the region where samples have been cached by the Perseverance rover. This means that the MSR rover needs to travel at great distances over very few Sols to meet mission requirements. This combination of requirements led to a 1 m/s travel speed for Fast Traverse with goals of traveling 1 km in a Sol. To further enhance the system's ability to traverse terrain, the ability for the system to steer wheels independently to precisely navigate through obstacles would prove useful in expressing more diverse autonomy behaviors.

These speeds are high enough that typical passive suspensions of current planetary rovers may not be adequate. These suspensions rely on gravity to maintain constant contact with the ground and this contact is used for wheel odometry localization and helping to prevent the rover from getting stuck in terrain. This means something similar to automobile suspension is better equipped for traveling at speed, much like the Lunar Roving Vehicles (LRVs) of the Apollo missions [54]. A spring suspension system allows for greater ability for maintaining ground contact while traveling over features in the terrain at speed. The introduction this spring suspension also provides an estimation of the load applied to a wheel through measuring the deflection distance. This suspension system combined with the independent steering system should fulfill the need for the system to achieve these speeds and test path planning algorithms.

The needed modularity for the rover is achieved through the design phase to make components easily interchangeable, but the need for planning for unknown missions is much more challenging. This can be accounted for by leaving extra room for electronics, payloads and the like, but there will be limitations to what hardware the system can support. This means potential missions need to be considered for the rover. Seeing that the system is intended to primarily cover ground quickly and test path planning algorithms, it would be very beneficial to plan for potential payloads that can help accomplish that task. These could include adding the ability to survey terrain stability [55,56], calibrating internal measurement units (IMUs) while moving, collecting scientific samples via robotic arms, unique sensor arrays for the autonomy to work with, and many other potential functions. There is one glaring issue with these potential missions that conflicts with the need of the rover’s primary missions. Many of these payloads need to make contact with the ground, and some may need to do so while on sloped ground. This means there would be very little ground clearance for the rover which needs a high ground clearance to scale over obstacles more easily. Incorporating a telescoping mechanism into these payloads would cause the rovers compartment dedicated to these missions to either grow taller or

become an open top section. An open top is not acceptable due to the needed sensors of the system, like 3D lidar, GPS antennas, camera, inertial measurement units (IMUs), sun sensor, or any other sensors that would benefit from being near the robot's geometric, or gravitational center. The system growing taller is also not acceptable as this raises the overall center of gravity of the rover, and decreases the system's stability to traverse terrain at speed. This culmination of requirements and considerations led to independent actuating the rover's suspension to bring the scientific instruments closer to the ground. This actuated suspension system also contributes to the system's ability to keep level despite the ground it is on and also provides potential functionality to traverse more challenging terrain.

Through the proposed design method, these requirements created through the logical decomposition of the broad needs of this rover have now constrained the system to a point where geometric assemblies of the system can be synthesized. A system that has an actuated independent suspension system with independent steering and a hollow compartment to harbor future payloads. Several means of accomplishing these requirements were considered in the trade study through deliberation amongst the design team. Discussions led to the consideration that it would be beneficial to make the rover symmetric to simplify the omni-directional steering control and help reduce the need for spare parts. This led to a four wheeled design with all of the previously mentioned features this rover needs to accomplish the traversal goals. The logical method of actuated suspension with this platform geometry is through a four bar mechanism. The culmination of these requirements and desired features resulted in the paper sketch shown in the figure below.

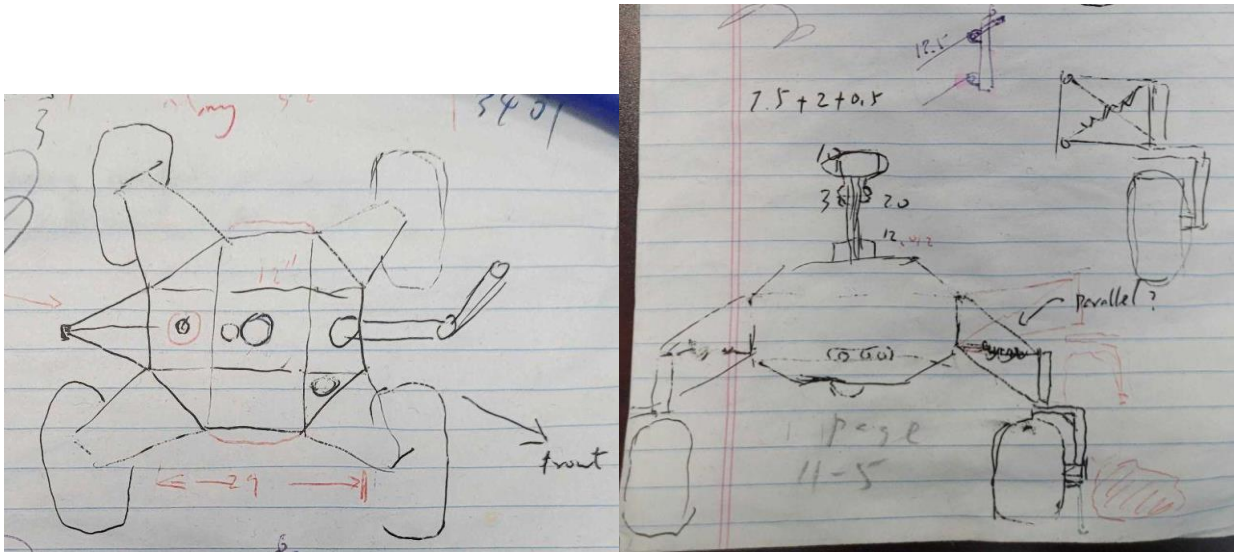

Figure 4.1: First Sketch of Fast Traverse Rover Top View (Left) and Side View (Right) by Yu Gu

Now that the robot's overall conceptual design has been determined, it is time to go through designing the subsystem components through the modeling-analysis loop. This was done progressively. An estimation of the components to be used was generated. This included

batteries, power management, computation, sensors, motors, and many other hardware components to fulfill the sketched design. Working from the top-down, the central frame was the first component to take shape. This primary structure determines how all of the other components work together, and quickly defines many details of the system. This component not only needs to take into account the technical requirements determined in the previous loop, but also the need for modularity, feasible assembly, ease of maintenance, simple manufacturing, and effective sensor placement. This criss-crossing body frame quickly took shape and analyzed for stresses according to weight estimates with healthy factors of safety. This assembly was sent out to be machined as soon as it was in a manufacturable state. The team's decision to have manufacturing and design occurring in parallel kept orders flowing to the shop throughout the project and allowed more time for prototypes of sections to be made.

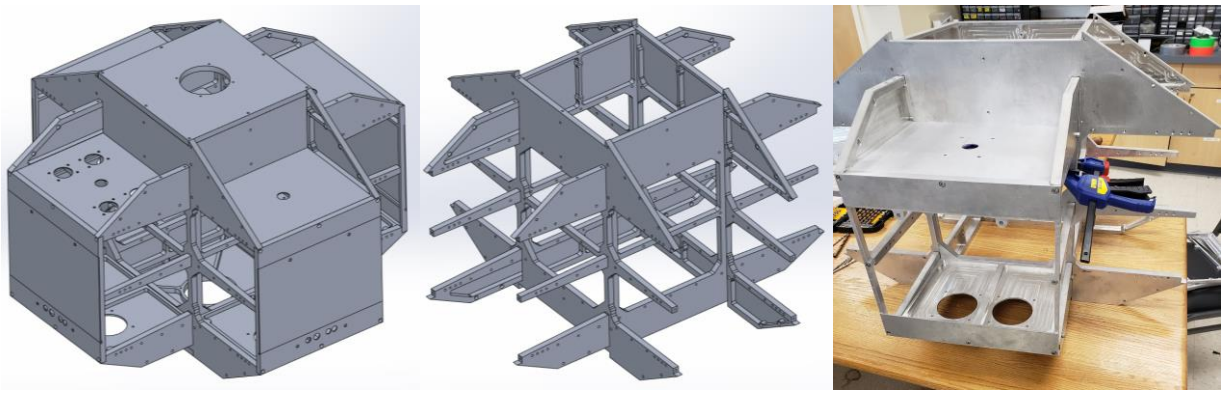

Figure 4.2: Body Assembly CAD (Left), Central Frame CAD (Middle), and First Frame Assembly (Right) by Dylan Reynolds

Since the frame design was solidified, the wheels were the next component to take shape. This logical jump between components was necessary. The drive assembly was the next component to determine the properties of other components down the design tree, like the steering and actuated suspension assemblies which interface the wheels to the body. Since this was a complicated form, a prototype was ordered to validate this assembly’s application to the rover. This was a thoroughly designed prototype and little revision was made between the version shown in the figure below and the final version used.

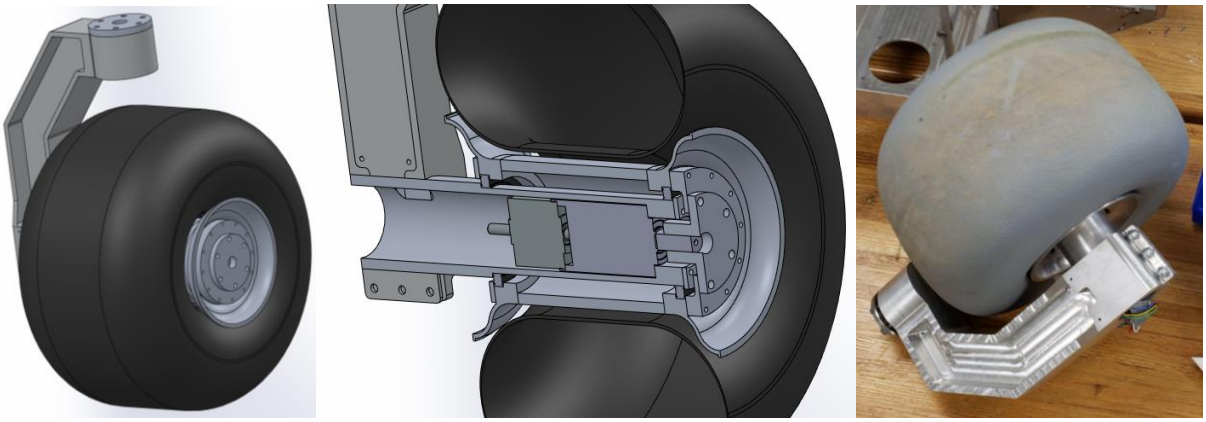

Figure 4.3: Wheel Assembly CAD (Left), Section View of CAD (Middle), and Wheel Assembly Prototype (Right) by Conner Castle

Then it was finally time the steering and actuated suspension took shape. These assemblies were all treated as one prototype to interface the components already made. It was clear from the previously established requirements that a spring suspension and a range of actuated travel for the rover to allow payloads to reach the ground were both needed.

The compact need of the steering assembly greatly restricted the off the shelf options available for this spring suspension and steering mechanism. The addition of the rover is estimated to weigh around 360 lb from current CAD models further reduced options. This spring system had to remain stiff and compact while being able to have a considerable amount of compression. A gear driven steering system is a simple solution to make room for applying sensors, but the assembly continues to be complicated by the spring assembly to take the load of the rover. The necessary addition of depth sensors and encoders further crowded the assembly.

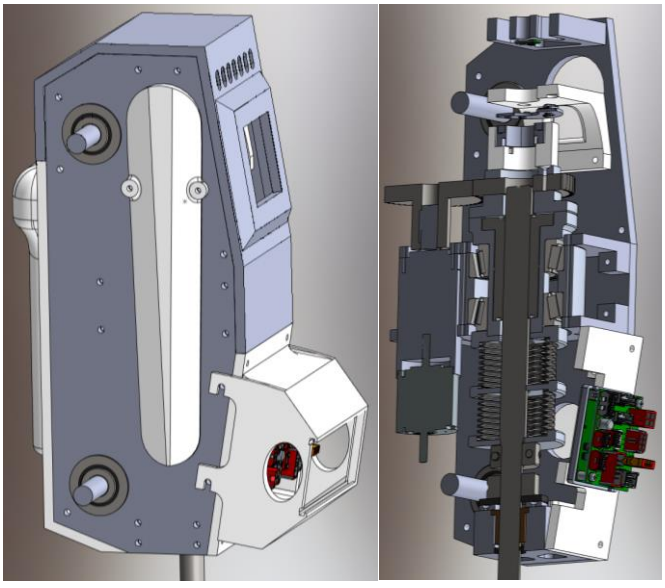

Figure 4.4: Steering Assembly CAD (Left) and Steering Assembly CAD Cross-Section (Right) by Chris Brindle

The steering assembly was designed in parallel with the actuated suspension system. This electrically driven four bar mechanism had many constraints already put in place by the three major subsystems already defined, but also partially determined the constraints of components in the steering assembly. The development of these two subsystems in parallel promoted accommodation of their needs in the design without diminishing performance.

The suspension subsystem shown in Figure 4.5 was designed to withstand half of the rover's weight on each leg, and achieve as much travel as possible. This maximized geometry was determined experimentally through computing the four-bar mechanism geometry in a MATLAB simulation made by Nick Ohi. This strange location led to adapting the mount for the linear servo.

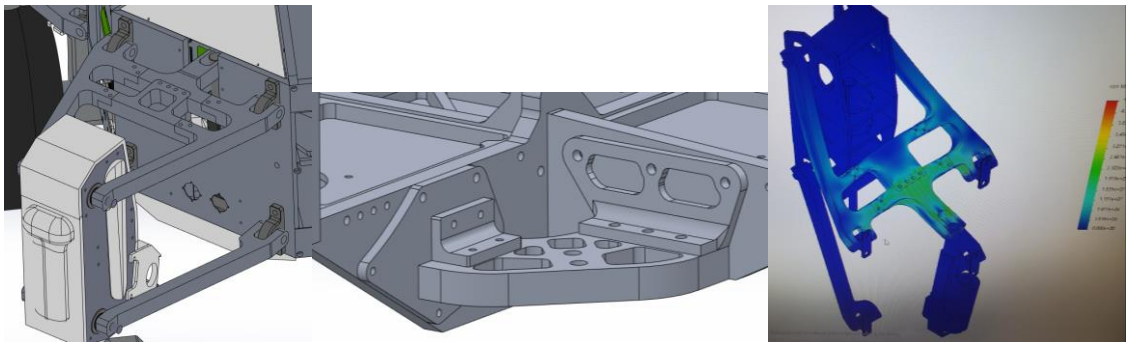

Figure 4.5: Actuated Suspension CAD (Left), Linear Servo Mount (Middle), Finite Element Analysis (Right) by Dylan Covell

These components were ordered to complete one "leg" of the rover. There were more minor revisions needed on this prototype as compared to the wheel assembly. After these quick adjustments, the final set of four steering and suspension assemblies were ordered and the CAD model of the rover was finally in a complete assembly.

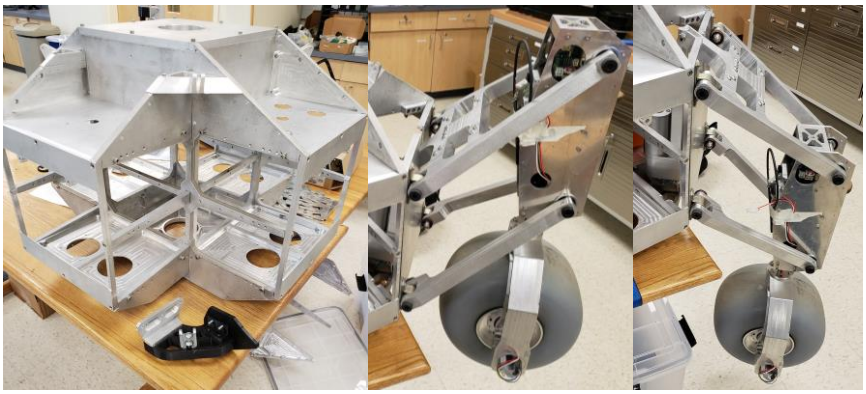

Figure 4.6: Body Assembly and Test Servo Mount (Left), Leg Assembly Raised (Middle) and Lowered (Right) by Dylan Covell and Chris Brindle

As seen in the figure below, many fine details and revisions were made throughout the design process. The geometric sketch from the early conceptual stage of this project did heavily inspire the form the system ended up taking. The feedback from the analysis of subsystem prototypes was a crucial feature of this design process.

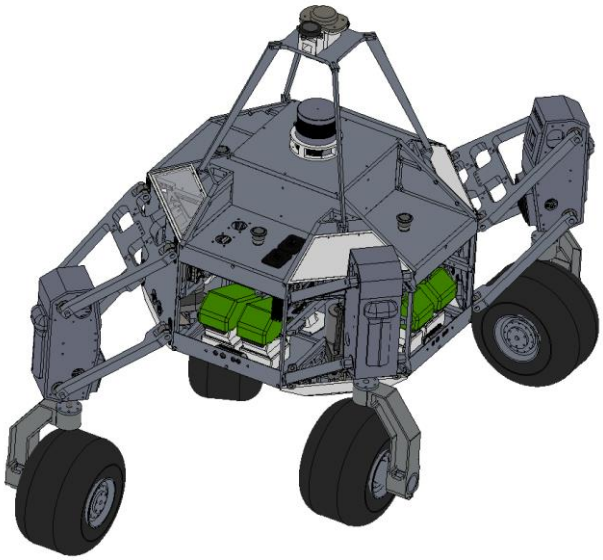

Figure 4.7: Completed Fast Traverse Rover CAD Model by Dylan Covell

Assembling and integrating the Fast Traverse Rover was a long endeavor. This was gradually done to ensure fitment of components, routing of cables, and testing of electronics. This rover's modularity was demonstrated through repeated assembly and disassembly to modify and test alternative components throughout this integration phase.

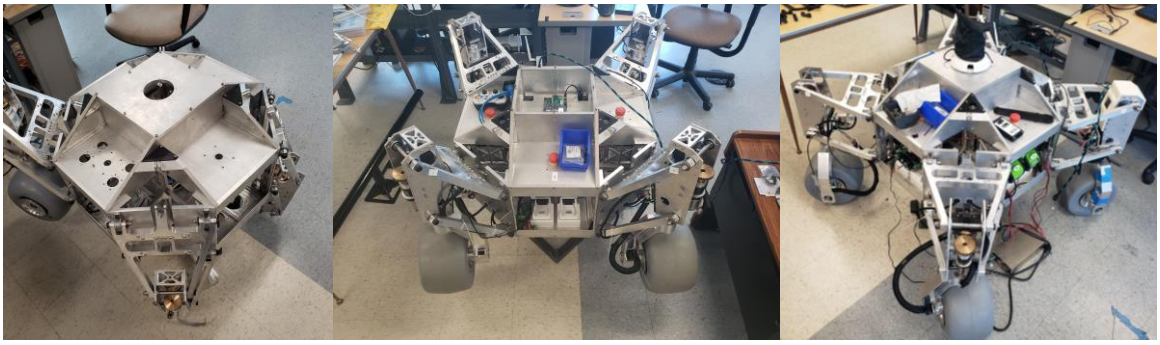

Figure 4.8: Fast Traverse Rover Suspension and Steering System Assembly by Eric Swanson, Jonas Bredu, and Dylan Covell

This rover's testing is still ongoing with the Interactive Robotics Lab at WVU. Sensors, controllers, and internal components have been exchanged to better accommodate the general purpose nature of this system. Current plans for testing focus primarily on validation of controlling the rover's drive and suspension system in the field. There are also plans to develop 3D printed tires to have better control over tuning the systems performance in field applications.

Thus far there have been no scientific payloads tested with the rover, but two have been prototyped so far. The automated shear vane apparatus made was a proof of concept to document soil bearing capacity without human testing, and would need heavy revisions to be applied to the Fast Traverse Rover [56]. The Zero-velocity Updates (ZUPT) has successfully provided a reference to calibrate an IMU while the system is moving, but this payload proof of concept would also need heavy revisions to function on the rover. [57, 58]

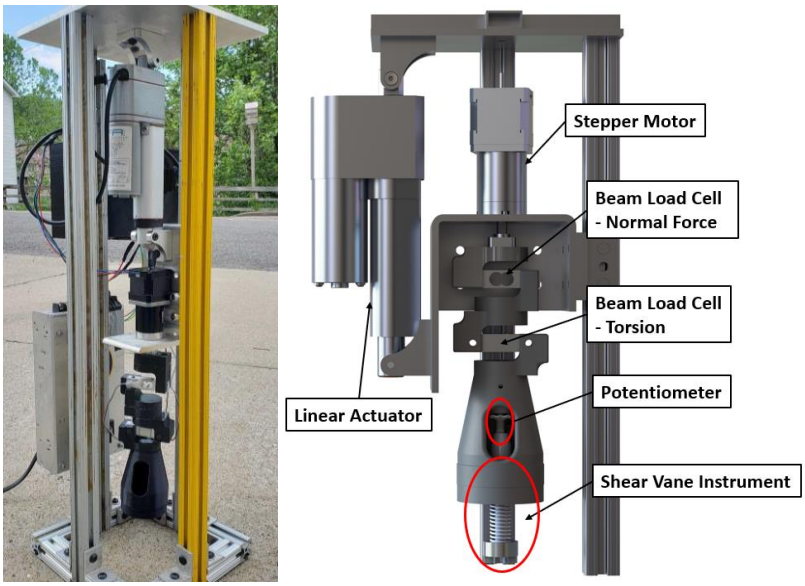

Figure 4.9: Automated Shearvane First Prototype (Left) and Version 2 CAD (Right) by Dylan Covell

Figure 4.10: ZUPT First Prototype (Left) and Version 2 CAD (Right) by Spencer Reigner

Figure 4.11: Fast Traverse Rover Test Drive (Left and Middle) and Current Status (Right) by Jonas Bredu, Jared Beard, Nick Ohi, and Dylan Covell

4.3. Results

This has been a long and arduous process to get the rover to this state. Fast Traverse will continue to progress through the time consuming integration and testing phase. There may be further alterations and improvements made as payloads work their way onto the rover. It is somewhat unknown how the project will progress in the future, but the application of the hybrid design process discussed in this paper has provided a good foundation for future exploration experiments at the Interactive Robotics Laboratory.

The biggest takeaway from this exploration of applying the proposed design method is the choice to develop subsystem designs and manufacturing higher up hierarchical systems in parallel. This did accelerate the project's progress, but the risk of subsystem's conflicting needs became evident at the interface of subsystems. The mounting of the linear servos for the actuated suspension required a rather unique assembly to accommodate the geometry needed to maximize suspension travel. The ability for the proposed design method to accommodate this conflicting decision within the development process is a limitation of the method. Applying this parallel design-manufacturing decision in the future requires significant experience and thoroughly thought out modular design to reduce the friction this decision introduces.

5. Case Study #2: Simple Robot System Design

5.1. Introduction

The WVU Interactive Robotics Laboratory (IRL), Field and Aerial Robotics (FARO) Lab, Navigation Lab, and Mining Department have developed a proof of concept mine surveillance system to automate safety inspections in cooperation with the Alpha Foundation. The system helps make this time consuming, highly repetitive, and dangerous job easier for human workers. The unmanned aerial vehicle (UAV) and unmanned ground vehicle (UGV) robot pair scan a mine's structure via Lidar mapping. The UGV navigates to surveillance points and the tethered UAV is deployed once stopped. [59] The UAV constructs a SLAM map of the structure and the feature recognition algorithm from the Mining Department documents the environment for easier processing by the operator [60].

This project has been the culmination of work amongst the involved research labs. The members involved with the UGV development are: Yu Gu, Dylan Covell, Jonas Bredu, Trevor Smith, Henry Vos, Nick Ohi, Chris Tastch, and Gio Molin. Their specific contributions are detailed in the design process below.

5.2. Design Process

This robot must survey mine support columns with LIDAR for signs of degradation. This objective puts very few constraints on the system accomplishing the mine surveillance task. Further decomposition is applied to better define the system's goals. These considerations include: what geometry of mine should this system survey, how much area does the system need to survey, and how long does the system have to survey the area. The test mine is a stone mine and consists of 12 meter tall columns that are about 70 meters wide. The inactive sections are to be inspected by this system, and these sections span approximately 2.5 km^2 of the mine. This system would need significant amounts of time to collect data on the region and additional time is needed for the system to navigate and return to the entrance of the mine. Additionally, the test mine is only available for inspection less than once a week. In order to maximize data collection, the development team decided that the system needs to operate for up to 8 hours at a time under the assumption that on average the system is traveling around 0.5 m/s during fully autonomous operations.

There are several potential system designs that can service this need. Among the design considerations in the trade study, two major system structures seemed appealing. An UGV with a telescoping boom could survey the mine, but this system structure could introduce significant deployment challenge. An UAV could survey the region quickly, but this system would not meet the operation time without a means of recharging. In the end, the design team decided to take on the UAV based system due to their prior experience with drone systems and the potential for UAVs to survey regions quickly. This added the requirement of incorporating a charging system into the design.

Further breaking down these requirements, it was clear that the 8 hour working period is too great for any one drone, and charging drones would result in more down time for the

hardware. Additionally the large area within the mine would result in increased time flying to locations from the deployment area than scanning the mine. The combination of these needs introduced a ground support vehicle to the system requirements to fill the performance gap. This better maximizes the drone's time scanning while in flight.

Despite the addition of the carrier, the UAV's total flight time was still tremendously limiting. There was a trade study to consider several ideas to extend the drone's operation time. These ideas included battery swapping, wireless charging, tethered power, and even employing several drones to scan the area. The battery swapping is very precise and complicated to do reliably. The wireless charging cannot charge the battery fast enough to sustain the flight of the drone over the time given. Using several drones does maximize the data collected in the given time, but this complicates system coordination with the UGV. The tether is a complicated control problem, but ended up being the most practical for our application due to its ability to provide the UAV power as long as the UGV was operational and that the ceiling's height makes the tether a manageable length. This design decision also helped to simplify the system by reducing the number of potential UAVs in the system to one.

This repetitive loop of logical decomposition, defining technical requirements, researching potential solutions, and conducting trade studies has gradually defined the major system features. This left the supporting features to facilitate the operation of this hybrid multi-agent system. This system needs to operate in the mine and lab conditions. The UGV and UAV should therefore have some tolerance to moist and dusty environments and fit into a standard freight elevator (48" wide). The UGV should possess a passive suspension system to maintain traction and traverse small obstacles. In this trade study, there are several considerations for the suspension. To further narrow the options, this suspension should focus on remaining strong and simple to make. This eliminated the options for independent steering and Ackerman steering in the suspension due to their added complexity. Passive rover suspensions for skid steering applications can become complicated rather quickly due to their mechanical connections. The simplest of these rover suspension systems is the split body design. Which relies on a single point of rotation for two rigid bodies. This highly desirable simplicity of the steering and suspension features in addition to the ability to easily harbor large amounts of equipment for a given volume are the factors that cemented this design layout for further development.

These features are all to be accomplished while remaining simple to manufacture, assemble, and service. The multidisciplinary brainstorming sessions rapidly progressed through the trade studies of the first loop in the proposed design process and defined a concept of operations. These features are further recognized and defined in the modeling-analysis loop of the hybrid design process. To simplify the reading flow of the remaining steps, we will be focusing on the design of the UGV.

The second stage of the proposed design process focuses on further defining the ground robot to the point of a manufacturable model. To begin, a geometric model accounting with all of the major subsystem components is constructed. These components were recognized by breaking down the robot into major subsystem modules, and selecting parts to fulfill the capabilities

outlined in the first loop. The drive system required motors that are compact and powerful enough to propel a heavy robot. Powering these demanding motors and other electronics requires large amounts of power. To simplify the power management of the system, the power supply can match the voltage of the motors. This 24 volt system would be supplied from an array of 12 volt batteries, and these batteries dominated the UGV's geometric and weight needs. This estimated weight driven by component selection in turn drove the needs of the motors. This feedback loop ceased when the theoretical battery capacity matched the power needs to operate all electronics for the 8 hour period. The computer, sensors, work lights, and other electronics needed to operate a rover system were picked from what the lab was most familiar with to further accelerate integration.

The rover's form prioritized interfacing these initially picked components. The first geometric model was purely a translation of the previous requirements with estimations of the dimensions and weights in hand. This sketch was developed into a geometric model in the Computer Aided Design (CAD) workspace with these key features and subsystems to further refine part placement.

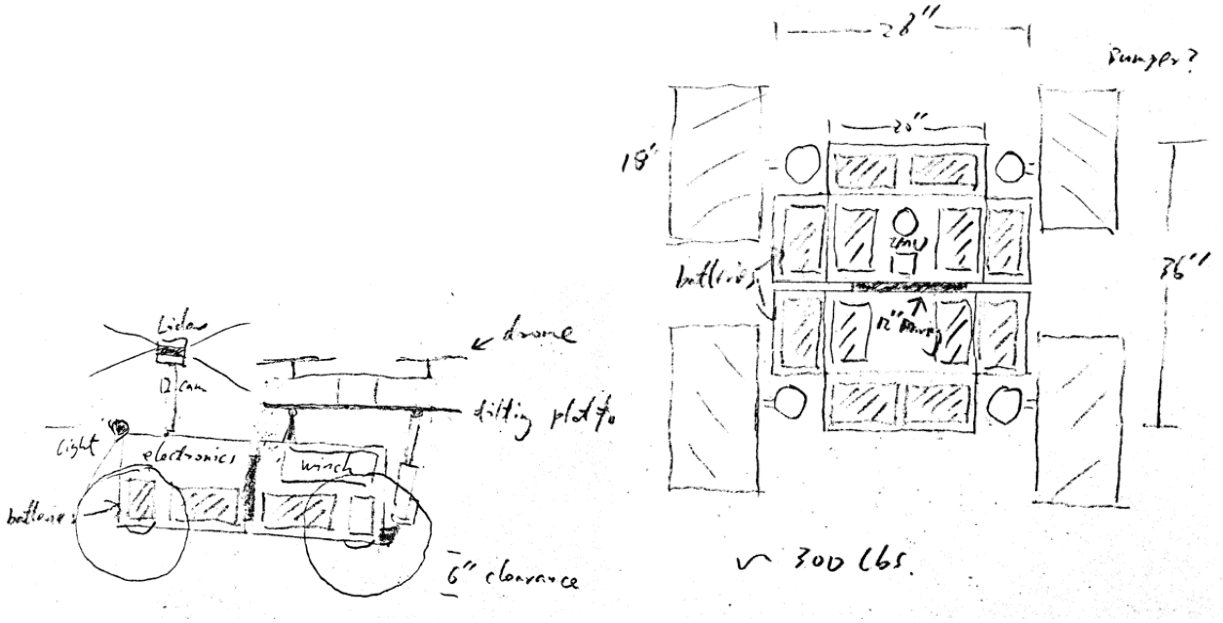

Figure 5.1: Initial Sketch of the Surveillance System by Yu Gu

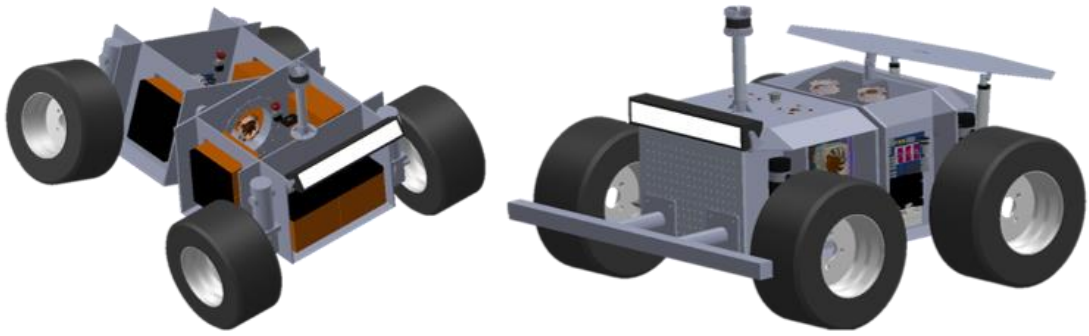

Figure 5.2: Geometric Model Iteration 1 (Left) and Iteration 2 (Right) by Dylan Covell

This cycle of refining the CAD model continues first with focus on adding more hardware and then structural detail to the model. This gradual process aims to incorporate all anticipated components and planning for future expansion while considering cable management, fastener placement, and how to assemble the components.

Analyses are conducted with Finite Element Analysis (FEA) to decrease mass where possible and validate the designs ability to facilitate operation in several worse case scenarios. Multidisciplinary studies of the systems feasibility were conducted in parallel to system synthesis with emphasis on manufacturability and serviceability. The design from all of these inputs resulted in the manufacturable model seen in Figure 5.4 below.

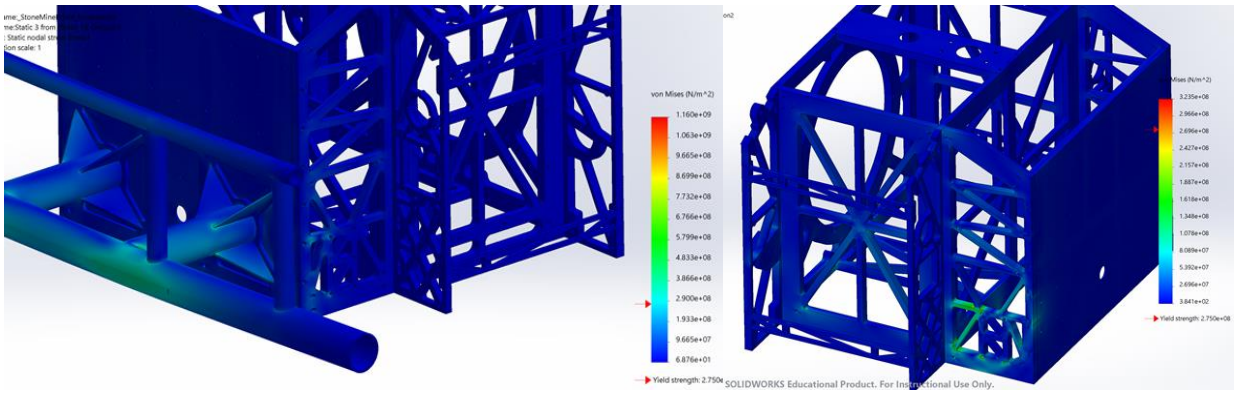

Figure 5.3: Visualization of FEA Analyses in Solidworks by Gio Molin

Figure 5.4: Manufacturable Model by Dylan Covell, Trevor Smith, and Gio Molin

The completion of the manufacturable model and sufficient digital validation of the systems to fulfill the technical requirement has signaled the transition to the manufacturing, integration, and testing phase. A majority of the custom mechanical components were waterjet thanks to the consideration of manufacturability in early steps. This combination of easy to make parts with fastening methods that use other parts as reference ensures good fitment. Further

integration of electronics with this simple chassis quickly filled the internal volume and leaving room for additional components in earlier steps made installation and rearrangement of components in these compartments manageable. This is the resulting operational prototype of the UGV, called “Rhino”, is shown in Figure 5.5 below and concludes the first pass of the proposed design methodology.

Figure 5.5: Initial Prototype Assembly by Dylan Covell and Jonas Bredu

Testing and integration of this system continued past this initial assembly. Reassessing the task and the technical requirements with more knowledge of the system greatly benefited the maturity and qualities of the system. This bottom-up feedback led to new components being added or exchanged to better achieve those goals. A stronger power management system was introduced to handle the current draw of the motors. The original power management solution buckled under the load of the motors. Motors with a higher gear ratio were acquired to better scale slopes under the system’s weight while still achieving reasonable speeds. The addition of charging ports for the batteries, improved camera systems, and a smaller computer motherboard were quality of life improvements implemented in this stage. The modularity of the UGV system made this iteration of subsystem components manageable.

Figure 5.6: Integrated UGV Prototype by IRL Lab, Jonas Bredu, Dylan Covell, Henry Vos, and Chris Tatsch, UAV Prototype by FARO Lab, Bernardo Martinez, Rogerio Lima, and Jeremy Rathjen

5.3. Results

Currently the Rhino UGV and "OxPecker" UAV are to continue testing at the mine facility. This testing will provide feedback to further mature the system performance. Future integration of the two systems will bring the project's original goal to reality.

As seen from the proposed design process detailed thus far, the Rhino UGV has matured very quickly over the course of its project life. This repetitive assessment and validation of the system have emphasized where Rhino needed improvement. Modeling and calculations only capture so much detail in their evaluation of the design. A lot of learning is done through these prototypes. Alterations to the project's qualities and definitions are a sign of experience. This proposed design method supplies opportunities to conduct this reassessment at times where significant experience has been accrued. Future applications of this method should strive to complete their first prototype before moving on to this reassessment phase to better maximize the improvements made between iterations.

6. Case Study #3: Simple Robot System Design and Bottom-up Control Software

6.1. Introduction

The WVU Interactive Robotics Laboratory (IRL) has developed a swarm-of-one platform nicknamed “Loopy”. The objective of this system is to study bottom-up methods in design and bottom-up methods of swarm control. These two concepts are heavily related in the sense that both definitions of bottom-up take in environmental stimuli to guide the decision making process. Whether that stimuli is driving a design decision, or the action a robot is to take next. It is difficult to begin designing a bottom up system from scratch. As a result, a modular design utilizing repetitive units and a bottom-up control software enable the formation of simple designs. This robot system is composed of 36 Dynamixel servos configured in a 2D closed loop. Although these servos are all controlled from a single computer, these servos operate as if they can only communicate with their adjacent neighbors. This allows each servo to function as independent agents with only local interactions. A decentralized system relies on environmental and agent interactions to determine system behavior. This exploration of swarm interactions in design is realized through shape matching. Bottom-up shape matching relies on agreement between units to operate effectively and reach the global goal shape provided by the user. This case study focuses on an extremely simple system design to provide more focus on the software side of robotic system development.

6.2. Design Process

This project began with the goal of exploring the control of decentralized swarms with simple connections. It was clear from this requirement that the mechanical system needed to be composed of many simple robots. Single degree of freedom (DOF) robots are a perfect fit for this simple agent requirement. These servos need to gather information to feed the control algorithm with external influences. The sensor that first comes to mind are absolute encoders for position feedback. There is also a need for some means of monitoring the load on a servo due to the rigid connections and risk for lack of cooperative movement occuring. These servos also need to be relatively easy to work with in order to promote progress in the system’s development. These three criteria were sufficiently fulfilled with the Dynamixel servos with ample sensors and software support.

System structure of these servos also needed to remain simple to keep the software control simplified and in focus. This decentralized swarm system can better demonstrate its potential when the structure is difficult to control with typical inverse kinematics and mathematical modeling. This complex agent interaction while remaining overall simple in structure had several designs considered. A 2D sheet of hexagons, like graphene, would be very difficult to control with inverse kinematics, but its overall construction quickly became too complicated as the idea was explored. An alternative and much simpler solution was found with

a simple closed loop. This closed 2D chain effectively puts all of these servos in parallel and in series. This complex interaction is even more difficult to generate inverse kinematic models of when there are many redundant joints in the loop. Thirty six of these servos in this loop fulfilled the simple construction and simple interactions requirement while providing a system structure that benefits from decentralized control.

There needs to be an experiment, or achievable goal, to apply this decentralized and rigidly connected swarm. There are several options considered for this case study. One promising experiment focusing on locomotion on a non-linear friction surface. This gives ample opportunity for unintended behaviors to emerge, but has many risks revolving around that friction performance. Another experiment option is shape matching. This has been an experimental application for robotic swarms before [61], and this is relatively lower risk due to relying on the servos encoders and integrated load sensors. The user provides a goal shape for the system to pursue in this case. This experiment assumes that the agents have perfect communication with their neighbors and that they are synchronized. It is also assumed that all agents within the swarm operate with the same rules and can only communicate with their direct neighbors in the loop. This results in a single agent only being aware of its own angle and load while communicating with adjacent units the error of their angle compared to the potential goal shapes. This swarm experiment quickly gained favor due to its comparative simplicity and is explained further in this paper.

There are many approaches to governing swarm behaviors, but the rigid agent interactions in this 2D loop applied to the shape matching experiment made swarm consensus a strong solution. These agreement protocols promote cooperation between agents to reach the goal shape and reduce the strain on the servos. The decentralized agreement protocols considered are relatively simple in nature. They can consist of taking the difference between two values amongst the agents in discrete time or a differential equation for continuous time. This discrete time agreement protocol makes the most senese due to this system does not prioritize speed and takes time to think and process information. However, the average of these differences are taken to apply this concept for multiple inputs.

The overall design and initial pursuit have now been solidified through the deliberation of the hardware and software requirements according to the original pursuit of studying the interactions of a decentralized robotic swarm with rigidly connected bodies. This resulting concept of operations for the project concludes the first and second loop of the proposed design method.

6.2.1. Hardware

The next loop of the proposed design process focuses on manufacturing and testing the system. The general model of this system is a 2D loop of 36 servos. The Robotis Dynamixel XL430-W250-T robotic servos are the preferred choice for this project due to their relatively low cost when compared to others. These are all mechanically linked together via Dynamixel brackets and wires to keep assembly simple. These servos require a 12V power supply, and one

with sufficient wattage to supply all 36 servos at peak was chosen for this system. All of these servos are being controlled by one computer through U2D2 controllers. The testing of this 2D loop, nicknamed “Loopy”, determined that only 30 of the 36 servos were controllable with one controller in this assembly. A second controller was added to the system to resolve this issue. The cables connecting to the controllers and power supply were lengthened to give the robot room to operate.

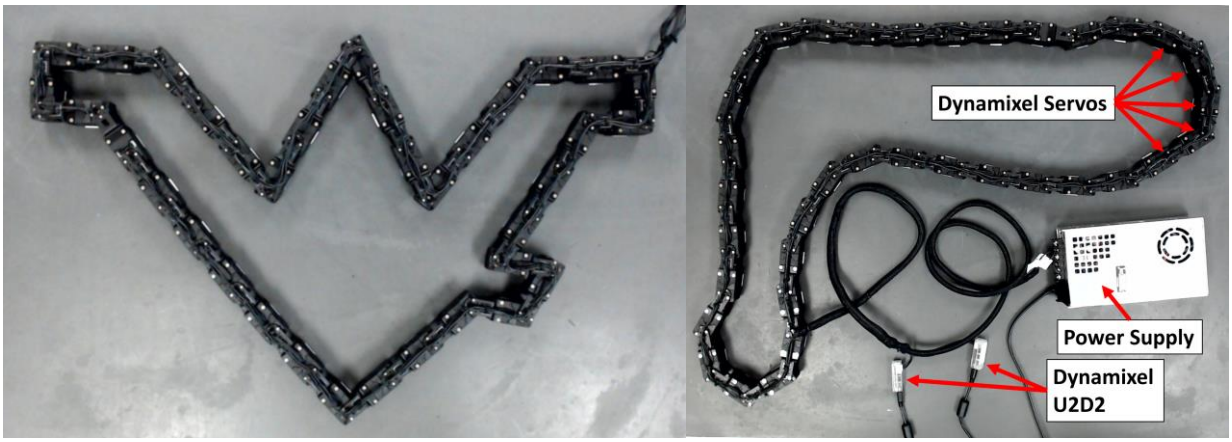

Figure 6.1: Configured in an Example Goal Shape (Left) and Complete System (Right)

6.2.2. Hardware Interface

The manufacturing and testing loop continues with interfacing the hardware with the computer and creating an easy means of testing and conducting experiments. MATLAB is the programming language of choice to keep software development and data processing simple. Initial control was achieved through examples and functions provided by the Dynamixel SDK package enabling communication with the Robotis U2D2 [62,63]. Control for experiments are done through the MATLAB Graphical User Interface (GUI) made for this robot through MATLAB App Designer. There are several features considered for this GUI to provide adequate feedback for troubleshooting and ease of conducting experiments. There needs to be a live feed of relevant data from all servos at a glance to help the user keep track of each servos state. The important values to monitor are the angle and load of the servo. The load is a measure of how many amps the servo is consuming and estimating the force being applied by the servo as a ratio of the maximum amperage these servos are set to consume. There also needs to be an easy way to manipulate the system by the user to promote initial experimentation and testing before algorithms are implemented. This is achieved via writing a goal angle to each servo and a switch to “turn on, or off” the servo. Dynamixel calls this setting torque. This setting simply means that when the servo is “on” it goes to the goal position until it is overloaded. These servos are limp when in the torque “off” setting. These are the major initial features of the GUI and helped manual manipulation of the servos.

This manual experimentation with the servos is only a stepping stone to the implementation of the control algorithms. The code of these control algorithms utilize the

functions and commands learned through the manual control portion of the GUI. There are some essential features needed to support rapid experimentation and data collection with these algorithms in the future. An interpretation of the system's shape to provide feedback to the operator during testing is nice to have, but also makes screen captures of the GUI more beneficial. Being able to save the data of the decisions made by the control algorithms is essential to allow for interpretation of the data after the experiment. The ability to save new goal shapes is also beneficial for testing and experimentation to better validate the algorithm's performance. Being able to reboot servos is also needed, as this resets their overloaded state and restores operation. The other major feature of the GUI is to turn on, or off, control algorithms easily. For example, writing the goal positions directly from the goal shape can provide easy validation of the algorithm's ability to converge to a solution over time. This can be done with two switches, one is able to turn on and off this direct command and then the other for the control algorithm being tested.

The combination of these features outlined in this design of the GUI greatly enables testing the system prior to the implementation of the bottom-up controls and can help validate considerations that need to be made in the controls. There have also been several quality of life features to increase the rate of experimentation. This resulting GUI is shown in the Figure 6.2 below.

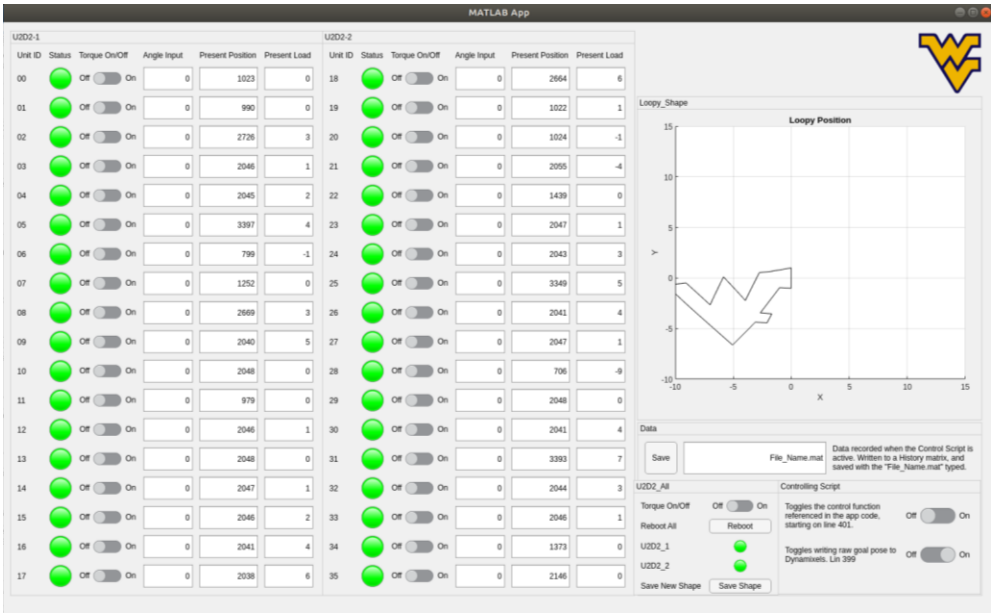

Figure 6.2: MATLAB GUI

6.2.3. Bottom-up Shape Matching Control Algorithm

The shape matching experiment entails that the system of independent servo agents are working together to progressively form the provided goal shape while minimizing the change in angle for the overall system. Breaking down this goal further, this means the system will

converge to the goal shape in different orientations due to the proximity to the goal angles. Each of these different orientations will be called formations. There are 36 possible formations for the system to select and act on. This shape matching task can be accomplished with the assumption that each agent is aware of all possible formations, its own angle in degrees (θ) and load as a percentage of the servo's maximum (γ), and is only able to communicate with its direct neighbors in the loop. Agents communicate their proximity to the 36 possible formations based on their own angle to neighboring units, called "Self Error".

Each servo measures their angle (θ) in degrees to determine the Present Position (P, θ^P) and compares it to all 36 Goal Positions (G, θ^G) in degrees. This difference is used as an error (ε) for each servo (K). This difference between the Present and Goal positions is called a "Self Error" (S, ε_K^S). This Self Error is the Present Position subtracted from the Goal Position and divided by 360 in Equation 6.1, a difference of angle with no units. This results in values ranging between 0 and 1, where 0 is a complete to the Goal Position. In this specific case, a Goal Shape is given by the user. While the Present Position is read from the servo at each time step.

Inputs:

Robot #: $K = [1, 2, \dots, 36]$ Goal Shape (Degrees): $\theta^G = [\theta_1^G, \theta_2^G, \dots, \theta_{36}^G]$

Present Position (Degrees): $\theta^P = [\theta_1^P, \theta_2^P, \dots, \theta_{36}^P]$ Present Load (%): $\gamma = [\gamma_1, \gamma_2, \dots, \gamma_{36}]$

Step 1: Self Error (ε_K^S) (36 Calculations per servo)

$$\varepsilon_K^S = \frac{[\theta_K^G - \theta_K^P]}{360} \qquad \varepsilon_1^S = \left[\varepsilon_1^{S(1)}, \varepsilon_1^{S(2)}, \dots, \varepsilon_1^{S(36)}\right] \tag{6.1}$$

These local observations of error are communicated to neighboring units. These errors are summed together to gather how well the cluster of three servos fit in the 36 possible solutions. This summation generates a "Local Error" $(\boldsymbol{L}, \boldsymbol{\varepsilon_K^L})$ for all 36 clusters around the loop in Equation 6.2. This Local Error is an estimate of the servo's global formation based on local information. The red circle on the left of Figure 6.3 below represents a single cluster of three servos. There are 3 clusters shown in the same figure below. Here 2 servos within a given cluster are observed to be shared between neighboring clusters.

Step 2: Local Error (ε_K^L) (36 Calculations per servo)

$$\varepsilon_K^L = [\varepsilon_{K-1}^S + \varepsilon_K^S + \varepsilon_{K+1}^S] \qquad \varepsilon_1^L = \left[\varepsilon_1^{L(1)}, \varepsilon_1^{L(2)}, \dots, \varepsilon_1^{L(36)}\right] \tag{6.2}$$

e.g. $\varepsilon_1^{L(15)} = \left[\varepsilon_{36}^{S(15)} + \varepsilon_1^{S(15)} + \varepsilon_2^{S(15)}\right]$

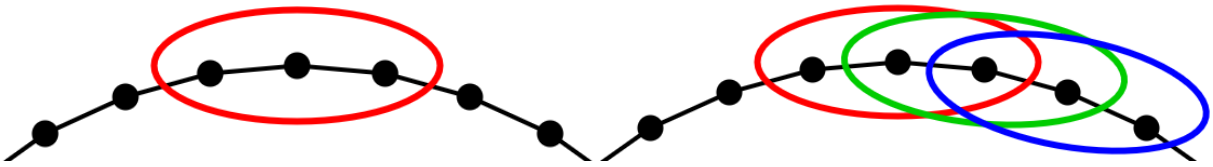

Figure 6.3: Single Servo Cluster (Left) and Three Servo Clusters in Loopy System (Right)

These units will use a decentralized consensus algorithm to reach an agreement of what is the best formation from these Local Errors. An average consensus algorithm is used for this application due to its ability to take in multiple information inputs and generate a single output [47]. This average consensus is generated each time step for 500 iterations. This number of iterations was determined experimentally. Where each servo takes the summation of these Local Errors (L, ε_K^L) for itself and its neighbors in a cluster of 3 to create an "Average Local Error" $(\underline{\boldsymbol{L}}, \underline{\boldsymbol{\varepsilon_K^L}})$ in Equation 6.3 below.

Step 3: Average of Local Errors $(\underline{\varepsilon_K^L})$ (36 Calculations per servo)

$$\underline{\boldsymbol{\varepsilon_K^L}} = \frac{[\varepsilon_{K-1}^L + \varepsilon_K^L + \varepsilon_{K+1}^L]}{3} \tag{6.3}$$

This Average Local Error is intended to replace the Local Error after each iteration. This Average Local Error is not applied to replace the actual Local Error until after iteration has concluded. This is to avoid giving bias to neighboring servos when they do their calculations, as these occur sequentially.

During these 500 iterations, the identity of which formation has the least error for each servo is documented as a means to track the system's performance. The "Iteration Number" (t) is a means of keeping track of the "Minimum Local Error" (I, I_K^t) from Equation 6.4 throughout these iterations.

Step 4: Minimum Local Error (1 Calculation per servo)

$$I_K^t = min\ \varepsilon_K^L \qquad I_K^t = [I_1^t, I_2^t, \dots, I_{36}^t] \tag{6.4}$$

A Target Position in degrees(T, θ^T) is selected for each servo after finishing these iterations and finally cataloging which formation to pursue. Equation 6.5 simply pulls the position from the provided Goal Positions (G, θ^G) according to the Minimum Local Error (I, I_K^t) and provides it for the servo to reference.

Step 5: Target Position (1 Calculation per servo)

$$\theta_K^T = \theta_{I_K^t}^G \qquad \theta_K^T = [\theta_1^T, \theta_2^T, \dots, \theta_{36}^T] \tag{6.5}$$

After declaring a Target Position (θ^T), a step is generated towards that Target from the Present Position otherwise called "New Position" (θ^{P+1}), in the logic below.This allows the servo to progressively get closer to the Target Position, and avoid overloading the system's servos. Currently a step is 100 ticks of the 4096 position encoder, equal to 8.8 degrees, or less.

Step 6: Generate Step Towards Target Pose (1 Calculation per servo)

if $\theta^T - \theta^P > 8.8$

$\theta^{P+1} = \theta^P + 8.8$
elseif $\theta^T - \theta^P < -8.8$
$\theta^{P+1} = \theta^P - 8.8$
elseif $\theta^T - \theta^P < 8.8$ and $\theta^T - \theta^P \geq 0$
$\theta^{P+1} = \theta^T - \theta^P$
elseif $\theta^T - \theta^P > -8.8$ and $\theta^T - \theta^P < 0$
$\theta^{P+1} = -(\theta^T - \theta^P)$
end

The system only moves to the New Positions once all are generated, as shown in Equation 6.6.

Step 7: Step Towards Target Pose (1 Calculation per servo)

$$\theta^P = \theta^{P+1} \tag{6.6}$$

Finally the system determines if it has met the conditions to reach a "Finish" state. This occurs when a servo has had the same Target Position for 10 time steps, and is tracked through a variable called "finish". When this occurs, the servo turns off its "Torque", or the ability for it to exert force. This essentially renders the servo passive, and any disturbance will cause it to turn back on and continue the process of stepping back towards its goal. This logic gate is represented in the block diagram figure below.

Step 8: Decide to "Finish"

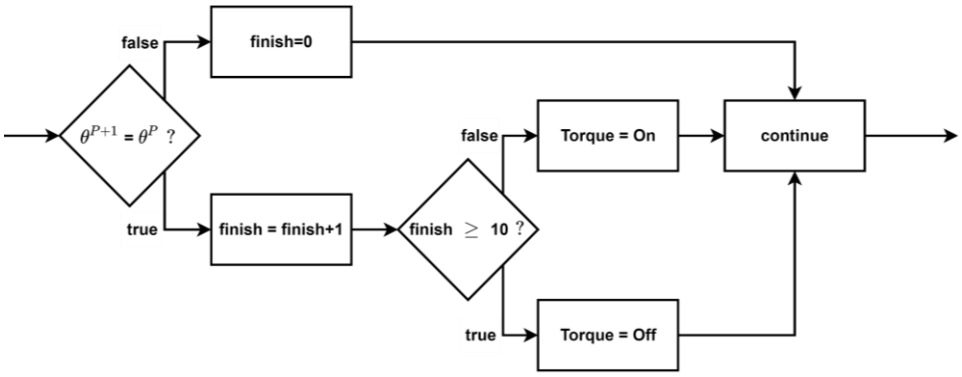

Figure 6.4: Block Diagram of Logic for Step 8

Now the steps outlined repeat indefinitely. This feature allows the system to reduce strain on the servos while continuously maintaining formation. This flow of the steps above is illustrated in Figure 6.5 below.

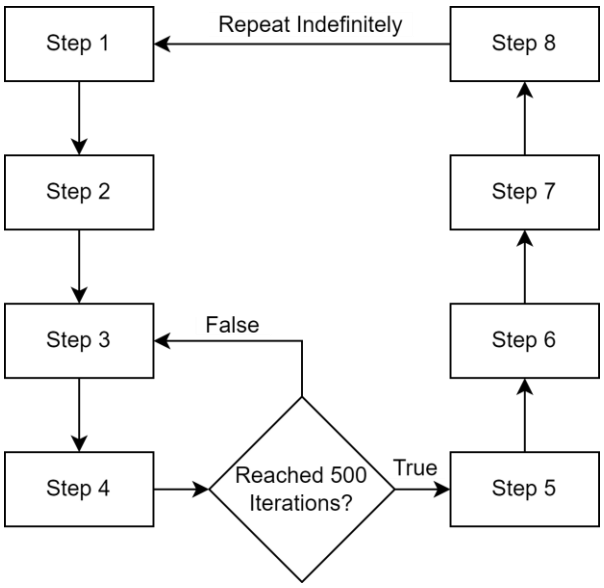

Figure 6.5: Block Diagram of Bottom-up Shape Matching Control Algorithm

6.3. Results

The hybrid design process utilized in this case study has provided an excellent means of determining a simple system and experiment to explore bottom-up control in application. This robot has demonstrated the ability to fulfill the needs of the shape matching experiment, and provide demonstration of swarm interactions in Figure 6.6. Further exploration of bottom-up control in robotics is needed due to the increased application of high degree of freedom systems, and systems with redundant degrees of freedom for increased robustness. This robot could stand to go through more iterations of refinement of the hybrid design method in future works to explore this method of control. Potential increasingly difficult bottom-up control applications include tasks like decentralized locomotion, environment exploration, or interactive design tool experiments.

In these future experiments, the proposed hybrid design method should be employed to reassess the structure of this swarm system. The repetitiveness of this method is necessary to establish properties that improve the quality and speed at which experiments can be conducted. These new iterations of the proposed method should progress quickly as compared to the first iteration shown in this case study. Future applications of this method should aim to have a well defined system objective to provide more direction to the definition of system requirements and efficiently narrow down potential solutions in trade studies.

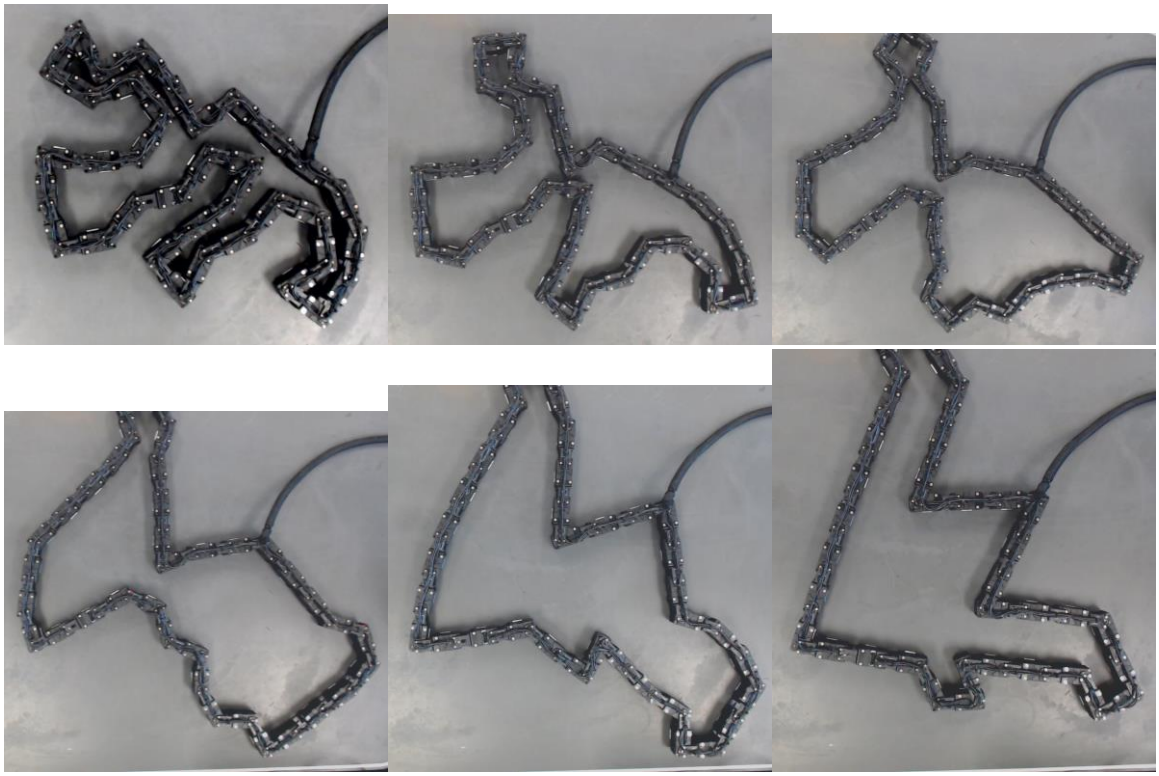

Figure 6.6: Progression from Random Start to Sample Goal Shape over 6 Steps, Step 1 Top Left to Step 6 Bottom Right

7. Conclusion

The proposed design method's flexibility has been demonstrated over several applications to real robots and has shown promising results for the small development teams seen in academia. In the first case study, the decision to apply the modeling-analysis loop in parallel with the manufacturing-integration loop did provide valuable feedback to complicated designs with risks. However, this also significantly complicated the further integration of subsequent systems and future application of this parallel method should only be applied to either narrowly defined systems, or projects with accelerated life cycles. The second case study emphasized the importance of iteration in product design. A significant amount of learning is accomplished through prototypes. The design team's experience is demonstrated through component additions and revisions conducted throughout a project. Future applications of this method will greatly benefit from completing a first functional prototype before beginning the next iteration. This time invested in the prototype will further inform and refine future works. The third case study showed the application of the proposed design method to the software side of robot design. This project did well to explore bottom-up control, but further improvements can elevate this system to a robotic design tool. Design teams employing this method in the future need to define their system objectives well in order to provide good direction to the projects' conception and therefore better narrow down potential solutions.

The culmination of this work has provided a few examples of systematic robot design, but this method is encouraged to be applied to more projects to further validate its effectiveness. For example, this design method was shaped by the experience from the case studies in this thesis and future application of this methodology should attempt to rigidly adhere to the proposed design method to validate its structure. Further studies may also include a scientific validation of this design approach through the application of this method by many separate design teams composed of diverse skill sets with the same problem. This experiment looks to see if teams produce similar results to validate this method's systematic design method and its ability to guide inexperienced engineers.

The application of this proposed design method in academia promotes the use of systematic design methods with clear periods of feedback. This incorporation of bottom-up input into the top-down design synthesis can improve the rate at which a system matures. The repetition of these design loops provides opportunities for the design team to apply lessons learned and encourages further removal of human bias from design. The work flow and examples of this method in this work leave little to be interpreted. This provides a comprehensible baseline for new engineers to work with. Continued practice of this method promotes the growth of inexperienced engineers in a way that better accommodates the needs of industry.

Extensions of this thesis can further apply the proposed design approach to robotic system design for more validation and verification of its methods. Additionally, the case studies seen in this thesis will continue to progress as part of their own projects. A future update of these developments guided by the proposed design process can provide a more complete picture of

these examples for inexperienced engineers and help validate the approaches explored in this thesis.

References

1. Roughgarden, J. (1979). Theory of Population Genetics and Evolutionary Ecology: An Introduction. Macmillan.
2. Crespi, V., Galstyan, A., and Lerman, K. (2008). *Top-down vs bottom-up methodologies in multi-agent system design - Autonomous Robots*. SpringerLink. Retrieved from https://doi.org/10.1007/s10514-007-9080-5
3. Böhringer, C., and Rutherford, T. F. (2007). *Combining bottom-up and top-down*. Energy Economics. Retrieved from https://www.sciencedirect.com/science/article/pii/S014098830700059X
4. Kahn, K. B. (1998). *Revisiting Top-Down Versus Bottom-Up Forecasting*. ProQuest. Retrieved from https://www.proquest.com/docview/226914424
5. Böhringer, C., and Rutherford, T. F. (2007). *Combining Bottom-Up and Top-Down*. Energy Economics. Retrieved from https://www.sciencedirect.com/science/article/pii/S014098830700059X
6. Sabatier, P. A. (1986). Top-down and Bottom-up Approaches to Implementation Research: A Critical Analysis and Suggested Synthesis. JSTOR. Retrieved from https://www.jstor.org/stable/pdf/3998354.pdf
7. Stokes, D., Matthen, M., Biggs, S., and Shea, N. (2017). Distinguishing Top-Down from Bottom-Up Effects. In *Perception and its modalities* (pp. 73–94). essay, Oxford University Press.
8. Fritz, B. R., Timmerman, L. E., Daringer, N. M., Leonard, J. N., andamp; Jewett, M. C. (2010). *Biology by Design: From Top to Bottom and Back*. Journal of Biomedicine and Biotechnology, 2010, 1–11. Retrieved from https://doi.org/10.1155/2010/232016
9. Birkhofer, H., Jänsch, J., and Kloberdanz, H. (2005). *An extensive and detailed view of the application of design methods and methodology in industry.* In DS 35: Proceedings ICED 05, the 15th International Conference on Engineering Design, Melbourne, Australia, 15.-18.08. 2005 (pp. 276-277).
10. Engineering Accreditation Commission. (2018). *Criteria for Accrediting Engineering Programs, 2019 – 2020*. ABET. Retrieved from https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2019-2020/#definitions
11. Haik, Y., Sivaloganathan, S., and M., S. T. M. (2018). *Engineering Design Process*. Cengage Learning.
12. Blessing, L., and Gericke, K. (2011). *Comparisons of design methodologies and process models across disciplines: A literature review* Research Gate. Retrieved from https://www.researchgate.net/publication/237049562
13. Howard, T. J.; Culley, S. J.; Dekoninck, E. (2008): Describing the creative design process by the integration of engineering design and cognitive psychology literature. In Design Studies 29 (4), pp. 160–180.

14. Lawson, B. (2006). *How designers think the design process demystified*. Elsevier Architectural Press.
15. Tomiyama, Tetsuo and Gu, P. and Jin, Yan and Lutters, Eric and Kind, Christian and Kimura, Fumihiko. (2009). *Design methodologies: Industrial and educational applications. CIRP Annals-Manufacturing Technology*. 58. 543-565. 10.1016/j.cirp.2009.09.003.
16. VDI. (1993). VDI 2221: Systematic Approach to the Development and Design of Technical Systems and Products. Beuth Verlag.
17. Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. TRW Defense Systems Group.
18. Forsberg, Kevin and Mooz, H.. (1992). *The Relationship of System Engineering to the Project Cycle*. Engineering Management Journal. 4. 36-43. 10.1080/10429247.1992.11414684.
19. Eckert, C. M.; Clarkson, P. J. (2005). The reality of design. In P. J. Clarkson, C. M. Eckert (Eds.): Design Process Improvement A review of current practice. London, pp. 1–29.
20. Gausemeier, J., and Moehringer, S. (2003). *NEW GUIDELINE VDI 2206 – A FLEXIBLE PROCEDURE MODEL FOR THE DESIGN OF MECHATRONIC SYSTEMS*. ICED23 - 24th International Conference on Engineering Design. Retrieved May 12, 2022, from https://iced.designsociety.org/publication/23949/
21. National Aeronautics and Space Administration. (2017). *NASA Systems Engineering Handbook*. NASA. Retrieved from https://www.nasa.gov/sites/default/files/atoms/files/nasa_systems_engineering_handbook_0.pdf
22. National Aeronautics and Space Administration. (2006). *NPR 7123.1 Systems Engineering Procedural Requirements*. NASA. Retrieved from https://nodis3.gsfc.nasa.gov/displayCA.cfm?Internal_ID=N_PR_7123_0001_andpage_name=main
23. National Aeronautics and Space Administration. (2006). *NPR 7123.1 - Chapter 3*. NASA. Retrieved from https://nodis3.gsfc.nasa.gov/displayCA.cfm?Internal_ID=N_PR_7123_0001_andpage_name=Chapter3#_Toc119904307
24. Gericke, K., and Blessing, L. (2011). *COMPARISONS OF DESIGN METHODOLOGIES AND PROCESS MODELS ACROSS DOMAINS: A LITERATURE REVIEW*. The Design Society - a worldwide community. Retrieved from https://www.designsociety.org/publication/30438/COMPARISONS+OF+DESIGN+METHODOLOGIES+AND+PROCESS+MODELS+ACROSS+DOMAINS%3A+A+LITERATURE+REVIEW
25. Atman, C. J., Adams, R. S., Cardella, M. E., Turns, J., Mosborg, S., and Saleem, J. (2013). *Engineering Design Processes: A Comparison of Students and Expert*

Practitioners. Wiley Online Library. Retrieved from https://onlinelibrary.wiley.com/doi/10.1002/j.2168-9830.2007.tb00945.x
26. Lehtonen, T., Juuti, T., Oja, H., Suistoranta, S., Pulkkinen, A., and Riitahuhta, A. (2011). *A FRAMEWORK FOR DEVELOPING VIABLE DESIGN METHODOLOGIES FOR INDUSTRY* . The Design Society. Retrieved from https://www.designsociety.org/publication/30439/A+FRAMEWORK+FOR+DEVELOPING+VIABLE+DESIGN+METHODOLOGIES+FOR+INDUSTRY
27. Boston Dynamics. (2015). *Introducing Spot Classic (previously Spot)*. YouTube. Retrieved from https://www.youtube.com/watch?v=M8YjvHYbZ9w
28. Boston Dynamics. (2016). *Introducing Spot (previously SpotMini)*. YouTube. Retrieved from https://www.youtube.com/watch?v=tf7IEVTDjng
29. Goldberg, D. E., and Samtani, M. P. (1986). *Engineering Optimization Via Genetic Algorithm*. cedb.asce.org. Retrieved from https://cedb.asce.org/CEDBsearch/record.jsp?dockey=0047877
30. Baldominos, A., Saez, Y., and Isasi, P. (2019). *On the automated, evolutionary design of neural networks: past, present, and future*. SpringerLink. Retrieved from https://link.springer.com/article/10.1007/s00521-019-04160-6
31. Weile, D. S., and Michielssen, E. (1997). *Genetic Algorithm Optimization Applied to Electromagnetics: A Review*. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/abstract/document/558650
32. Burger, S. P., Jenkins, J. D., Huntington, S. C., and Perez-Arriaga, I. J. (2019). *Why Distributed? A Critical Review of the Tradeoffs Between Centralized and Decentralized Resources*. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/abstract/document/8643507
33. Span.IO. (2022). *A Smarter Electric Panel*. Span. Retrieved from https://www.span.io/
34. Lumin. (2022). *Lumin : Responsive Energy Management Platform*. Lumin. Retrieved from https://www.luminsmart.com/platform/smart-electrical-panel
35. Rudman, R., and Bruwer, R. (2016). *Defining web 3.0: Opportunities and challenges*. The Electronic Library. Retrieved from https://www.emerald.com/insight/content/doi/10.1108/EL-08-2014-0140/full/pdf?title=defining-web-30-opportunities-and-challenges
36. Anderson, M. (2019). *Exploring Decentralization: Blockchain Technology and Complex Coordination*. Journal of Design and Science. Retrieved from https://jods.mitpress.mit.edu/pub/7vxemtm3/release/2
37. Kumar, V., Prorok, A., and Guerrero-Bonilla, L. (2017). *Formations for Resilient Robot Teams*. IEEE Xplore. Retrieved from https://doi.org/10.1109/LRA.2017.2654550
38. Kumar, V., Saldaña, D., and Guerrero-Bonilla, L. (2018). *Design guarantees for resilient robot formations on lattices*. IEEE Xplore. Retrieved from https://doi.org/10.1109/LRA.2018.2881231

39. Lim, Q. W. (2019). *How do multiplayer games sync their state? Part 1*. Medium. Retrieved from https://medium.com/@qingweilim/how-do-multiplayer-games-sync-their-state-part-1-ab72d6a54043
40. Lim, Q. W. (2019). *How do Multiplayer Game sync their state? Part 2*. Medium. Retrieved from https://medium.com/@qingweilim/how-do-multiplayer-game-sync-their-state-part-2-d746fa303950
41. Eklund, P., Rusinowska, A., and de Swart, H. (2007). *A consensus model of political decision-making*. Springer. Retrieved from https://link.springer.com/content/pdf/10.1007/s10479-007-0249-2.pdf
42. Ren, W., Beard, R. W., and Atkins, E. M. (2007). *Information Consensus in Multivehicle Cooperative Control*. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=andarnumber=4140748andtag=1
43. Tanner, H. G., Pappas, G. J., and Kumar, V. (2004). *Leader-to-Formation Stability*. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=andarnumber=1303690
44. Mesbahi, M., and Egerstedt, M. (2010). *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press. Retrieved from https://www.jstor.org/stable/j.ctt1287k9b
45. Li, Y., and Tan, C. (2019). *A survey of the consensus for multi-agent systems*. Taylor and Francis Online. Retrieved from https://doi.org/10.1080/21642583.2019.1695689
46. Castanedo, F., Patricio, M. A., Garcia, J., and Molina, J. M. (2007). *Bottom-Up/Top-Down Coordination in a MultiAgent Visual Sensor Network*. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=andarnumber=4425292
47. Kriegleder, M., Oung, R., and D'Andrea, R. (2013). *Asynchronous Implementation of a Distributed Average Consensus Algorithm*. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=andarnumber=6696598
48. Parker, C. A. C., and Zhang, H. (2009). *Cooperative decision-making in decentralized multiple-robot systems: The best-of-N problem*. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=andarnumber=4801702
49. National Aeronautics and Space Administration. (2022). *Mars Sample Return Campaign*. NASA. Retrieved from https://mars.nasa.gov/msr/
50. Corum, J., and White, J. (2014). *MARS CURIOSITY ROVER TRACKER*. The New York Times. Retrieved from https://archive.nytimes.com/www.nytimes.com/interactive/science/space/mars-curiosity-rover-tracker.html#sol665
51. National Aeronautics and Space Administration. (2022). *NASA's Perseverance Celebrates First Year on Mars by Learning to Run*. NASA. Retrieved from https://mars.nasa.gov/news/9134/nasas-perseverance-celebrates-first-year-on-mars-by-learning-to-run/

52. National Aeronautics and Space Administration. (2008). *Science Priorities for Mars Sample Return*. NASA JPL MEPAG. Retrieved from https://mepag.jpl.nasa.gov/reports/ND-SAGreport_FINALb1.pdf
53. Golombek, M. P., Otero, R. E., Heverly, M. C., Ono, M., Willford, K. H., Rothrock, B., Milkovich, S., Almeida, E., Calef, F., Williams, N., Ashley, J., and Chen, A. (2017). *CHARACTERIZATION OF MARS ROVER 2020 PROSPECTIVE LANDING SITES LEADING UP TO THE SECOND DOWNSELECTION*. Universities Space Research Association. Retrieved from https://www.hou.usra.edu/meetings/lpsc2017/pdf/2333.pdf
54. The Boeing Company. (1971). *LUNAR ROVING VEHICLE OPERATION$ HANDBOOK: CONTRACT NASB-25145*. NASA. Retrieved from https://www.hq.nasa.gov/alsj/lrvhand.html
55. Hedrick, G., Ohi, N., and Gu, Y. (2020). *Terrain-Aware Path Planning and Map Update for Mars Sample Return Mission*. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=andarnumber=9126145
56. Hedrick, G., Covell, D., and Gu, Y. (2020). *IN-SITU TERRAIN ANALYSIS FOR PLANETARY ROVERS*. West Virginia University. Retrieved from https://yugu.faculty.wvu.edu/files/d/432f8ea8-5dae-40e0-8665-19944d772d54/instrumentation_slip_istvs__owner_gabrielle_.pdf
57. Kilic, C., Gross, J. N., Ohi, N., Watson, R., Strader, J., Swiger, T., Harper, S., andamp; Gu, Y. (2019). *Improved Planetary Rover Inertial Navigation and Wheel Odometry Performance through Periodic Use of Zero-Type Constraints*. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/abstract/document/8967634
58. Kilic, C., Ohi, N., Gu, Y., and Gross, J. (2021). *Slip-Based Autonomous ZUPT Through Gaussian Process to Improve Planetary Rover Localization*. IEEE Xplore. Retrieved from https://ieeexplore.ieee.org/abstract/document/9387093
59. Samarakoon, K., Pereira, G., and Gross, J. (n.d.). *Impact of the Trajectory on the Performance of RGB-D SLAM Executed by a UAV in a Subterranean Environment*. ICUAS'22, 2022. Retrieved from https://controls.papercept.net/conferences/scripts/abstract.pl?ConfID=325andNumber=131
60. Bendezu de la Cruz, M. A. (2021). *Evaluation of LIDAR systems for rock mass discontinuity identification in underground stone mines from 3D point cloud data*. The Research Repository @ WVU. Retrieved from https://researchrepository.wvu.edu/etd/10243/
61. Nagpal, R., and Kavli, F. (2014). *A self-organizing thousand-robot swarm*. Wyss Institute. Retrieved from https://wyss.harvard.edu/news/a-self-organizing-thousand-robot-swarm/
62. ROBOTIS-GIT. (2016). *Robotis Dynamixel SDK*. GitHub. Retrieved from https://github.com/ROBOTIS-GIT/DynamixelSDK
63. ROBOTIS. (2018). *U2D2*. ROBOTIS. Retrieved from https://www.robotis.us/u2d2/