Carnegie Mellon University

Robotics Capstone Project

System Readiness Review

Friction Force Explorers:

Don Zheng
Neil Jassal
Yichu Jin
Rachel Holladay

supervised by
Dr. Cameron Riviere

Version 1.0
April 11, 2017

Contents

List of Figures

1 Build Progress

This section details system development and progress made since the last milestone presentation. Progress is split into three major sections: electromechanical, software, and integration. Electromechanical updates detail chassis build progress, as well as setup of the electronics to drive the motors for locomotion and using the writing implement. Software updates describe progress towards subsystem completion. Integration details updates made with regard to integrating the electromechanical and software systems, and testing functionality.

1.1 Electromechanical Updates

1.1.1 Electrical Updates

Electrical progress is separated into two main parts: connecting the motors to the motor controller and Raspberry Pi, and powering the motors and Raspberry Pi controller. Each of the motors was wired up to an Adafruit Motor Controller for easy use, which mounts as a shield on top of the Pi. For the time being, we have chosen not to connect the motor encoders. Given that we are using localization for motions, and vector directions for the robots are updated at every control loop iteration, we have hypothesized that localization will be enough to ensure accurate motion. The robots will never be moving more than a few inches without updated directional commands, making fine tuned encoder-based motor control unnecessary.

The battery packs for the Raspberry Pi and motors were directly connected to the respective pieces of hardware. Until the electronics mount is built, we have been placing the battery packs on the robot, or holding them during testing. We plan to attach them to the mount using velcro.

1.1.2 Mechanical Updates

As shown below in Fig.1, we have integrated the structures for holding the microcontroller and AprilTag. These two structures were built using laser patterned acrylic and bonded to the chassis using super glue. Funtionality and usability were tested during the locomotion and localization tests and proved promising. The AprilTag and the Raspberry Pi are currently taped to the supporting structures, but they will be mounted using Velcro tape instead in the future.

Besides integrating holding structures, we also mounted batteries to the chassis such that all the electronics were onboard. In the last milestone, we presented the idea of mounting these batteries underneath the AprilTag. However, doing so caused the AprilTag to vibrate during motion, which decreased localization accuracy. Instead, we mounted the batteries on the two sides of the robot. These batteries were mounted to the chassis via 3M VHB tape. For the ease of switching, we will use Velcro tape instead for the next iteration.

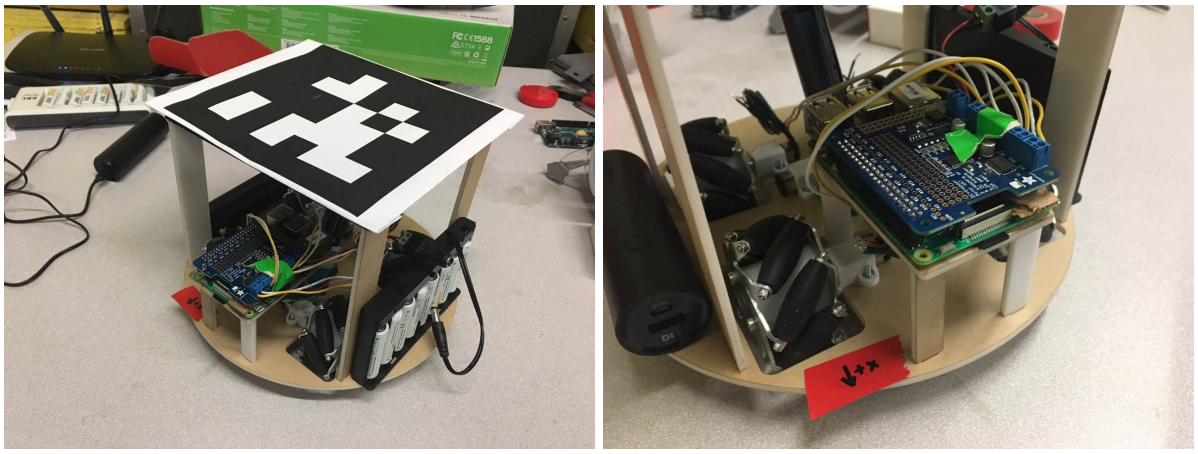

Figure 1: Prototype Overview

During the tests, one of the four 3D printed wheel adapters broke. This part has broken quite frequently. To address this issue, we are redesigning the piece to replace the existing adapters.

We have also designed the camera jig, which is shown below in Fig.2. It is constructed using 80/20 aluminum frames, which were chosen due to the frames' ease-of-use and durability. The jig has a footprint

of 6ft by 6ft and a height of 8ft. 8ft was determined to be the optimal height for camera mounting, to capture all corners of the 6ft by 6ft work space. The camera will be mounted to the jig using Velcro tape. Due to 80/20 structure's easy construction, we can simply disassemble the jig into two pieces for transportation. Therefore, despite its large size, the camera jig is not restricted to any specific space.

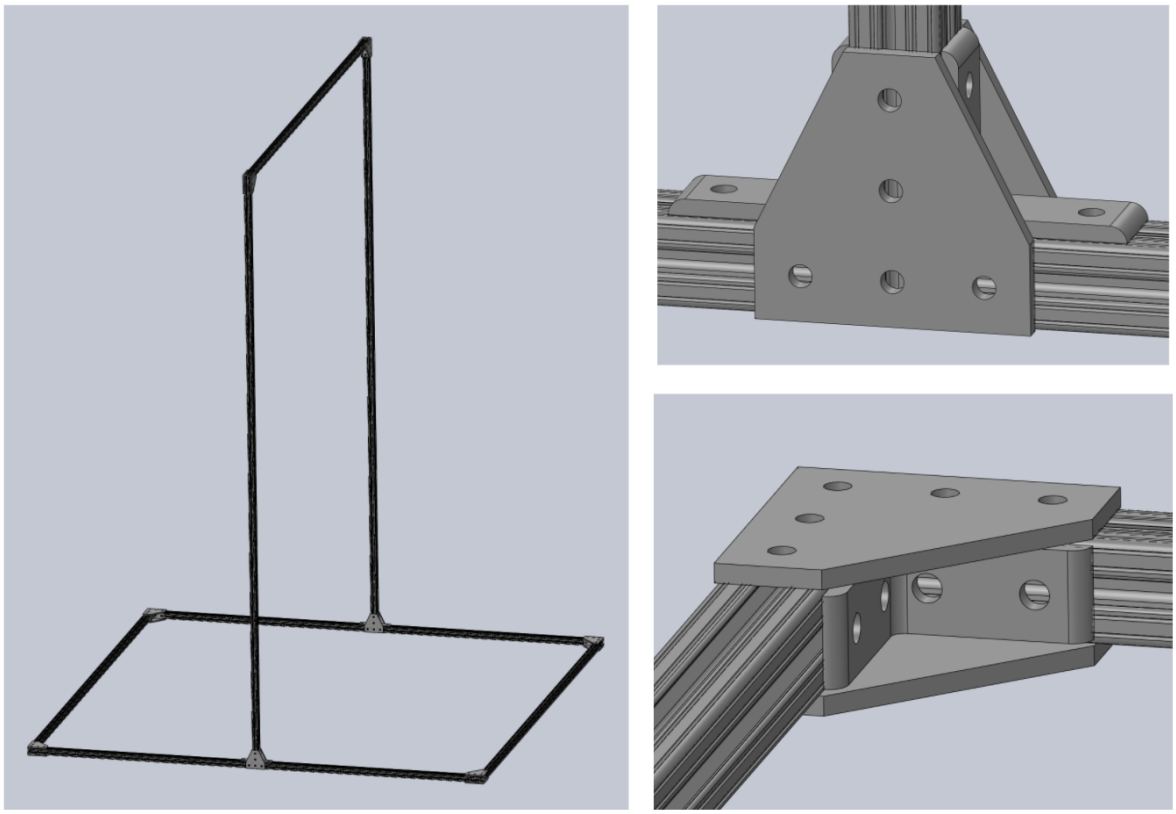

Figure 2: Camera Jig CAD

1.2 Software Update

We detail the software progress made across the following subsystems. Most subsystems have reached the point of usability, and at this point most additions enhance ease-of-use and functionality.

1.2.1 Locomotion

The locomotion subsystem has been fully implemented as a part of the onboard controller code. The subsystem is capable of determining motor commands based on a target vector direction to move the robot along the specified vector. The robots have mecanum wheels and can therefore move omnidirectionally. This fact, coupled with the fact that we plan to only have to move along fixed straight-line vectors as specified by the SDP subsystem, means the robots never have to rotate. The goal of locomotion is only to translate along vectors, and never rotate. However, implementation of the mecanum control equations includes the ability to have the robots rotate during operation. The main use of rotation will be to correct any rotational error detected by the localization subsystem.

1.2.2 Localization

The localization subsystem has been completed, and is successfully able to use a combination of the AprilTags library, and Boost Python to transmit position and orientation of each AprilTag back to the controller. The controller then computes an affine warp using the specified corner tags, and warps the coordinates into a fixed dimension space. For example, if the input space is from coordinates (0,0) to (10,10), the controller will warp the space from pixel coordinates to the (0,0), (10,10) frame. While this can potentially warp the image being drawn as it stretches to accomodate a fixed input space, the change in dimensions is small enough not to affect output quality. Orientation of the robots is also computed,

and will be sent to the robots to correct any rotational error. Fig.3 shows an annotated test image of the six AprilTags to be used, with their respective labels and tags marked for visualization.

Figure 3: Annotated AprilTag testing setup

1.2.3 Scheduling, Distribution and Planning (SDP)

Before SDP integration, we have formatted our planner to take input in the form of a standarized message type. Additionally, we added flexibility to the planning system by allowing inputs of arbitrary dimension. Given an input of size M by N the planner will re-scale the drawing to match the size of the drawing surface, X by Y. Following these updates the current planner was integrated into main code base. We will be testing and adding on-board collision prevention soon.

1.2.4 User Interface

We completed development of a UI that allows users to draw the lines they would like the robots to complete. The interface is shown in Fig.4. The user drags their mouse to draw a series of lines. The user has the option to clear their current drawing, allowing them to start over. Once the user is finished they can input the filename under which they would like to save the drawing and exit the tool. The tool records the line drawing and saves it to our database. We envision using this tool to create our own drawings and to add an interactive element to the demo.

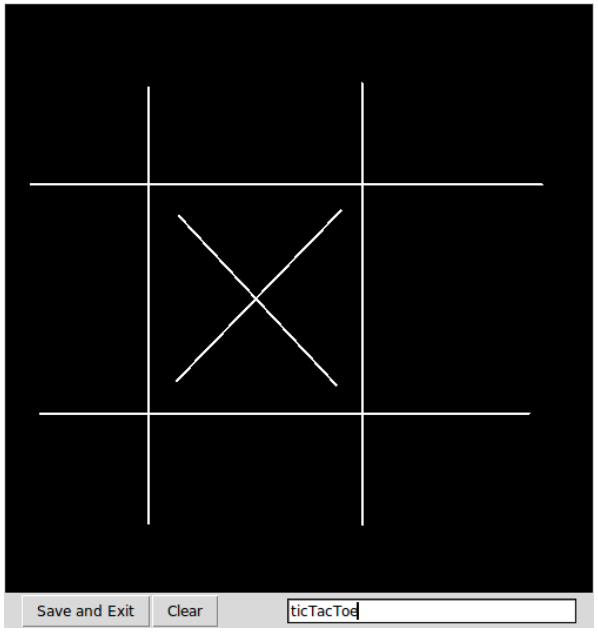

Figure 4: Drawing Interface for Inputting Requests

1.3 Integration Updates

System integration has involved assembling the electronics - including the motors and Raspberry Pi controller, and attaching them correctly to the mechanical system. For now, we were able to combine the locomotion subsystem with the onboard controller to send vector commands to the motors. The robot is able to move omnidirectionally and correct for rotation, however without localization it is impossible to detect rotational error. After running some simple motion tests without the encoder, we found that the robot was well within our positional accuracy requirements, and we do not believe the encoders will be necessary. Once localization is connected to the onboard system, we will be able to confirm that the encoders are not necessary.

Next steps involve integrating the communication and localization systems. Current tests used the onboard system only to run locomotion commands. The first task is to enable offboard communication to send locomotion commands to the robot. Once consistent and accurate communication is established, localization will be added. Using localization, we can begin running simple plans that move the robot from point to point within the designated drawing space.

Parallel testing will involve the writing implement. Now that the robots can move individually, testing and improvements to writing while drawing will start. Tests of writing quality during various motions and writing speed limits will be done to ensure we can meet quality and consistency requirements for the final drawing.

2 Project Management

2.1 Work Breakdown Schedule

In this section, we present the Work Breakdown Schedule for the project.

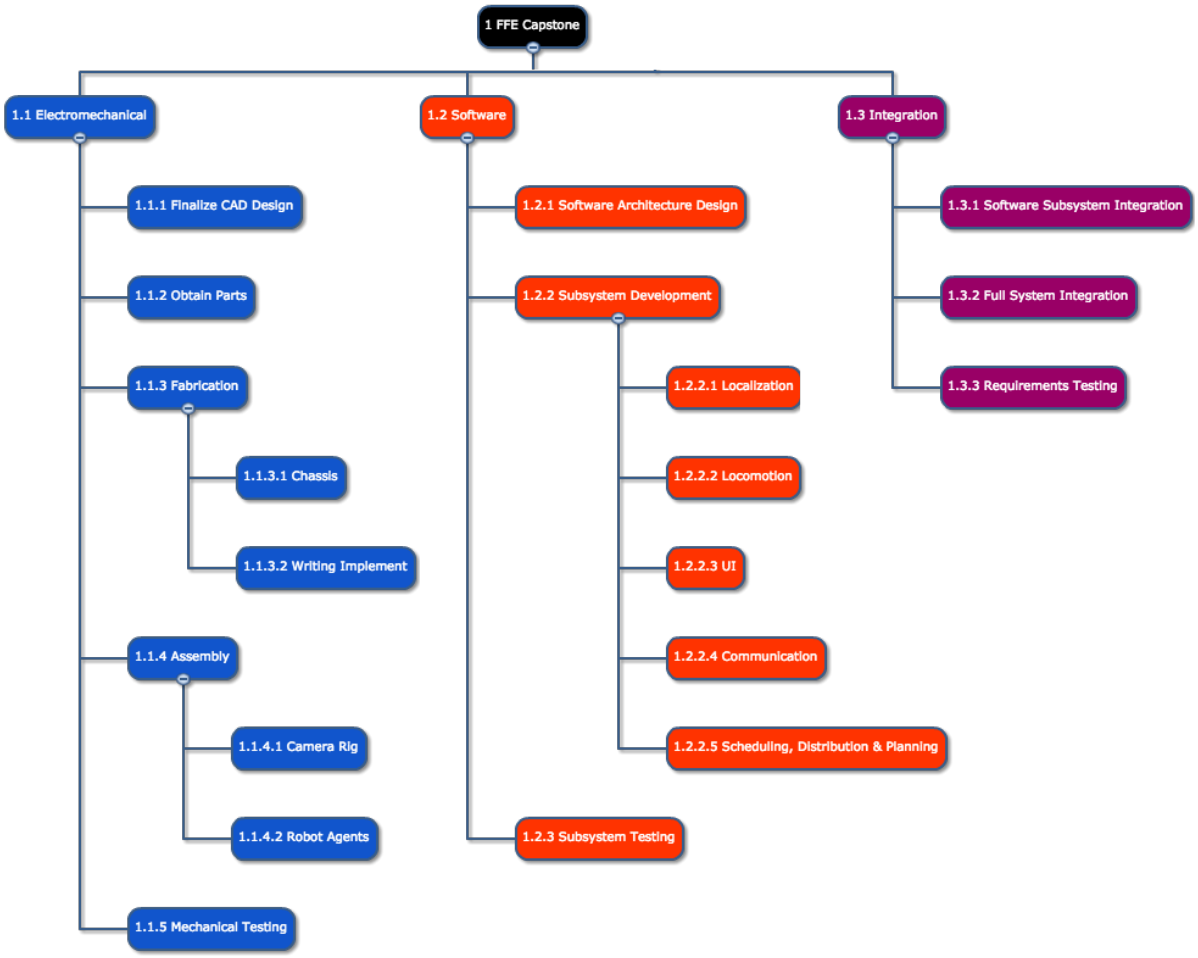

Figure 5: Full WBS for the project

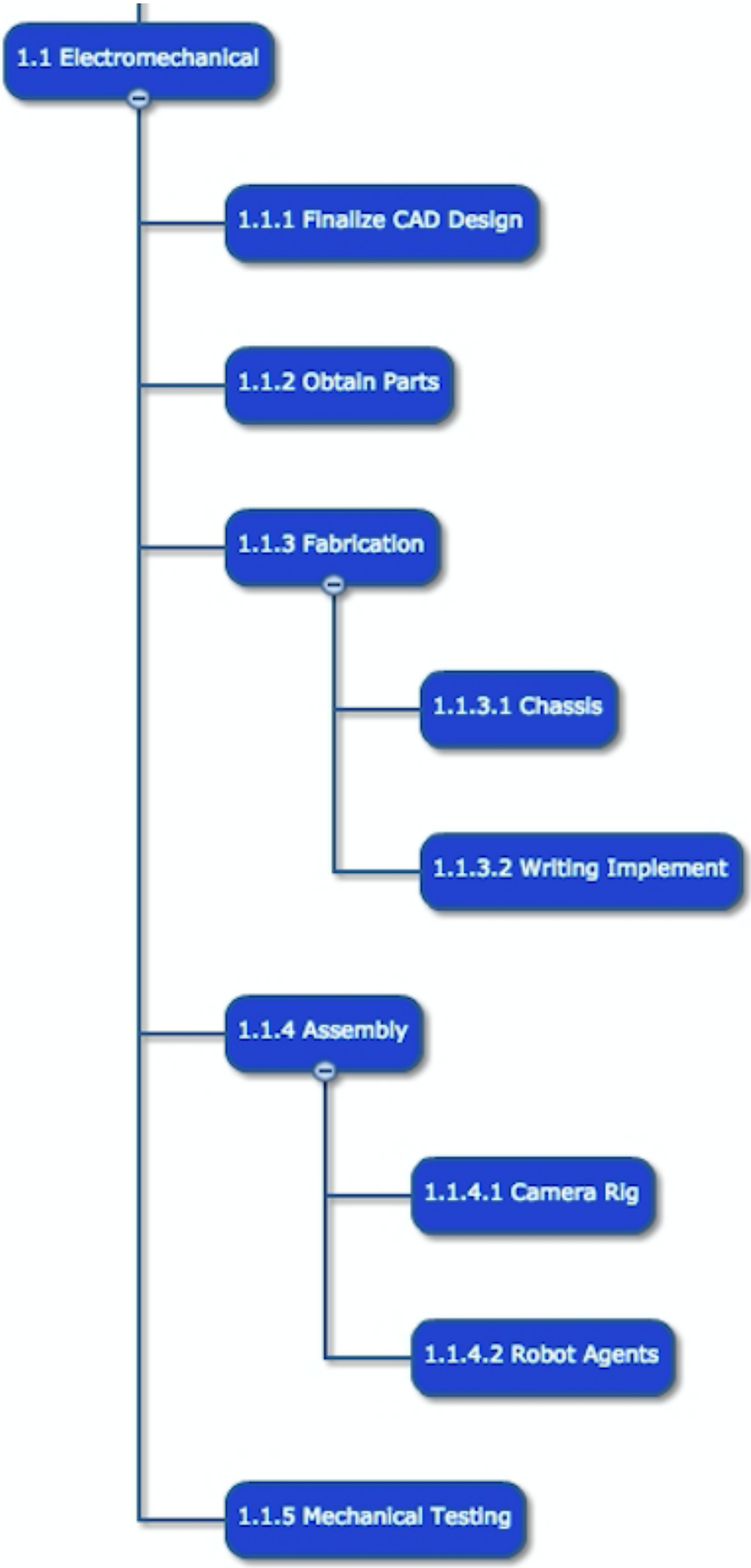

Figure 6: Electromechanical WBS section

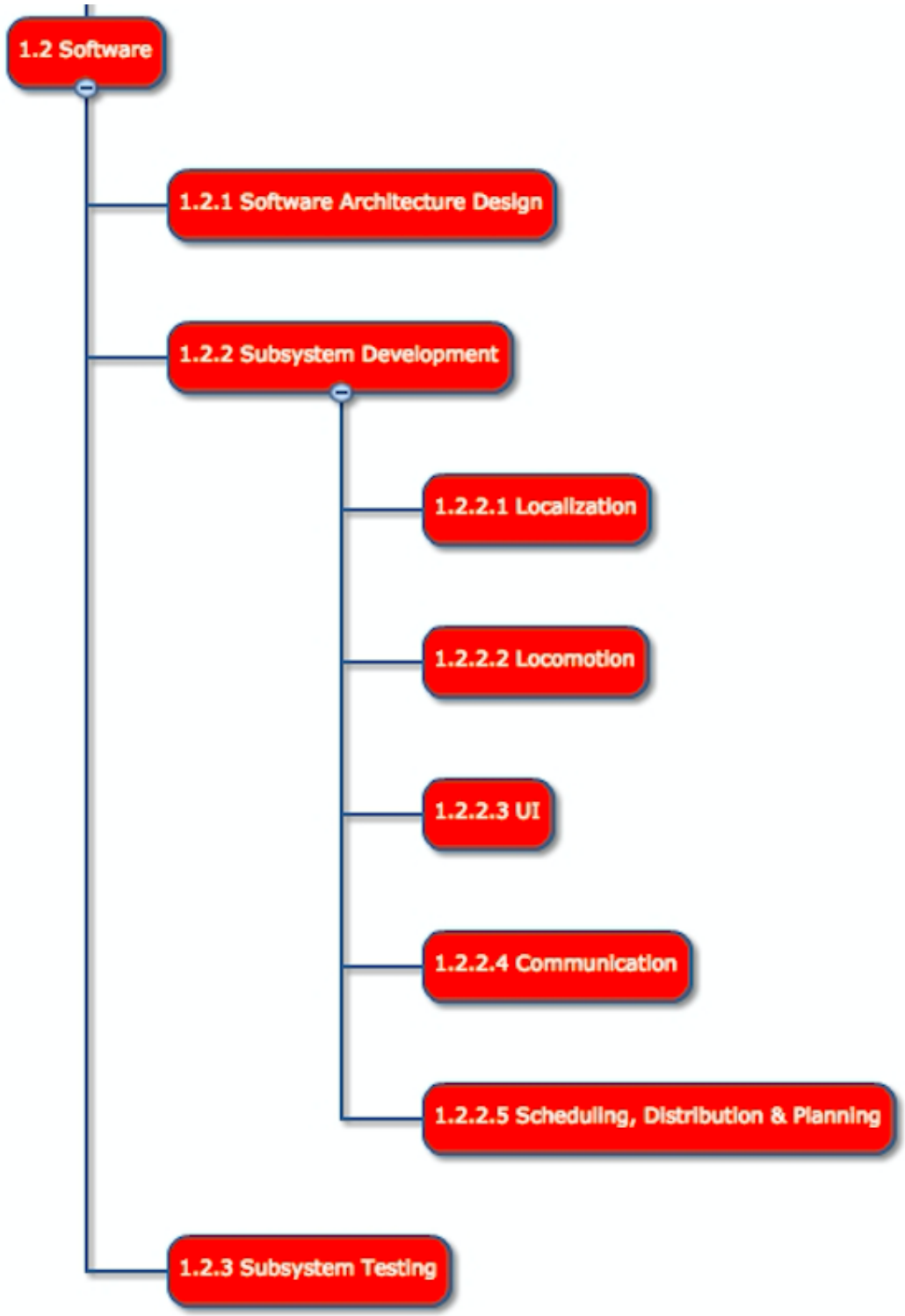

Figure 7: Software WBS section

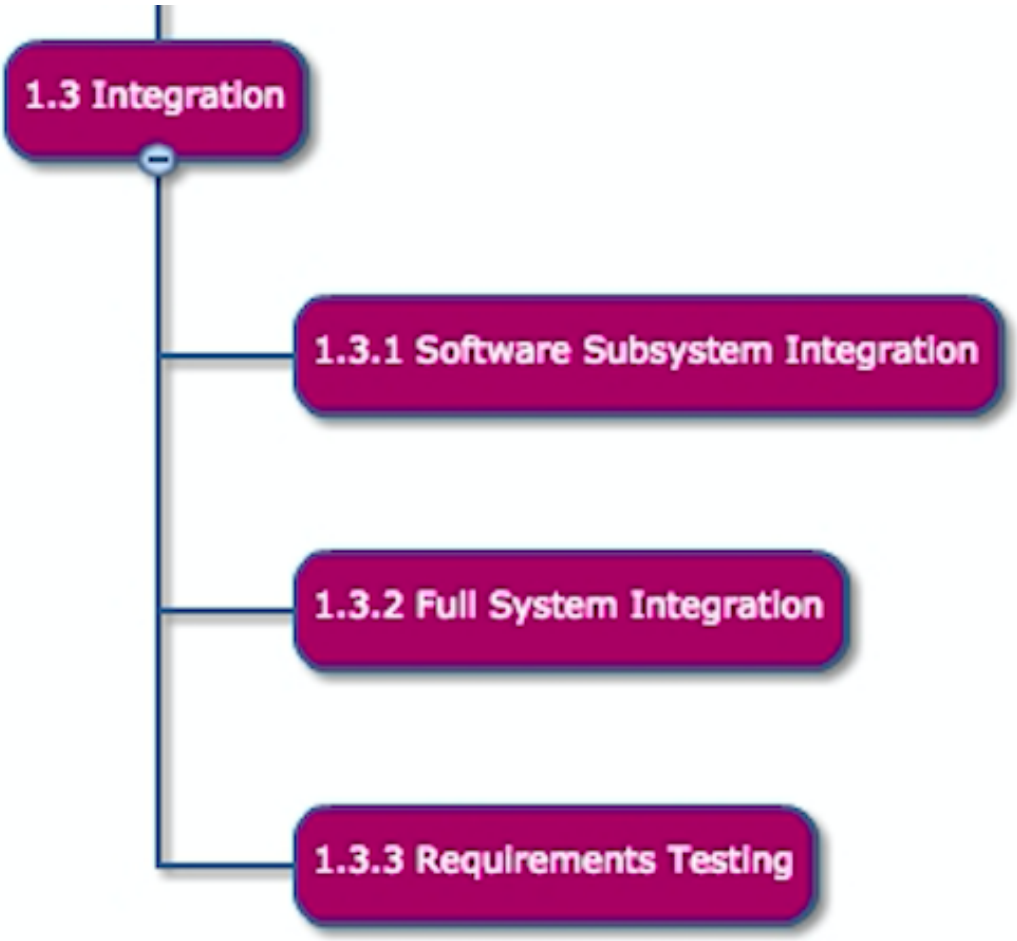

Figure 8: Integration WBS section

The WBS dictionary entries include more information on each of the work elements of the project. Information such as estimates for the amount of time each task will take and their dependencies will help us adhere to our schedule, while determining the owner of each task will improve tractability of the workflow.

WBS#:	1.1.1	Task:	Finalize CAD Design
Est. Effort (hrs):	3	Owner:	Eric
Resources:	CAD software	Work products:	CAD files
Description:	Update the CAD design to fit the final design of the robot		
Input:	Previous designs, new design ideas and requirements		
Dependencies:	Complete design review		
Risks:	Designs cannot be completed on time		

WBS#:	1.1.2	Task:	Obtain Parts
Est. Effort (hrs):	20	Owner:	Don
Resources:	Parts list	Work products:	Parts order receipt
Description:	Finalize the parts list, and contact the necessary people to ensure that parts are ordered		
Input:	CAD designs, electronics designs		
Dependencies:	Finalized CAD designs		
Risks:	Parts ordering procedure is more time consuming than expected		

WBS#:	1.1.3.1	Task:	Fabricate Chassis
Est. Effort (hrs):	4	Owner:	Eric
Resources:	CAD designs, MechE shop	Work products:	Chassis components
Description:	Use the Mechanical Engineering machine shop to fabricate components necessary to build the chassis		
Input:	CAD designs, parts		
Dependencies:	Obtain parts		
Risks:	Machine shop is not available, injury from operating machines		

WBS#:	1.1.3.2	Task:	Fabricate Writing Tool
Est. Effort (hrs):	4	Owner:	Eric
Resources:	CAD designs, MechE shop	Work products:	Writing tool components
Description:	Use the Mechanical Engineering machine shop to fabricate components necessary to build the writing implement		
Input:	CAD designs, parts		
Dependencies:	Obtain parts		
Risks:	Machine shop is not available, injury from operating machines		

WBS#:	1.1.4.1	Task:	Assemble Camera Rig
Est. Effort (hrs):	3	Owner:	Don
Resources:	Scrap wood	Work products:	Camera rig
Description:	Build the rig used to hold the camera for the vision system above the drawing space		
Input:	Measurements from demo space		
Dependencies:	Confirmation of demo space location		
Risks:	No extra wood is available, demo space does not have adequate room for the camera rig		

WBS#:	1.1.4.2	Task:	Assemble Robot Agents
Est. Effort (hrs):	5	Owner:	Eric
Resources:	Tools, fasteners	Work products:	Two robot agents
Description:	Use fabricated components to build the two robot agents in the system		
Input:	Fabricated components		
Dependencies:	Fabricate chassis and fabricate writing tool		
Risks:	Parts are broken during assembly, extra parts or fasteners are needed		

WBS#:	1.1.5	Task:	Mechanical Testing
Est. Effort (hrs):	3	Owner:	All
Resources:	Tools, fasteners	Work products:	Two robot agents
Description:	Perform mechanical testing on the robots in accordance with our testing guidelines		
Input:	Mechanically complete robots		
Dependencies:	Assemble robot agents		
Risks:	Tests are failed, and significant time or extra resources are needed to correct the tests		

WBS#:	1.2.1	Task:	Software Arch. Design
Est. Effort (hrs):	3	Owner:	All
Resources:	None	Work products:	Function headers
Description:	Design function I/O, and create function headers for all files we will use in the robot		
Input:	Software flowchart, decisions on software libraries		
Dependencies:	Complete design review		
Risks:	Selected software libraries have compatability issues		

WBS#:	1.2.2.1	Task:	Localization Subsystem
Est. Effort (hrs):	6	Owner:	Neil
Resources:	AprilTag library	Work products:	Working localization
Description:	Fill in the function headers for the localization system to develop an end-to-end localization solution for the robots		
Input:	Function headers and design for localization system		
Dependencies:	Software architecture design		
Risks:	Localization system or library is unable to perform to expectations		

WBS#:	1.2.1	Task:	Locomotion Subsystem
Est. Effort (hrs):	5	Owner:	Don
Resources:	Adafruit Motor controller library	Work products:	Control system for motors, robust motion model
Description:	Create a complete set of functions that can be used to direct the robots around the workspace		
Input:	Software flowchart, decisions on software libraries		
Dependencies:	Software architecture design		
Risks:	Interfacing issues with motors, damanged electronics hardware, unreliable motion models		

WBS#:	1.2.1	Task:	User Interface Subsystem
Est. Effort (hrs):	4	Owner:	Rachel
Resources:	Various UI libraries	Work products:	User interface including calls to other subsystems
Description:	Create a visually appealing and intuitive user interface for the robot system		
Input:	Software flowchart, decisions on software libraries		
Dependencies:	Software architecture design		
Risks:	Libraries are not available		

WBS#:	1.2.2.4	Task:	Comunication
Est. Effort (hrs):	8	Owner:	Neil
Resources:	Wireless comm. libraries	Work products:	Functions for sending info. back and forth from robots
Description:	Create a reliable communication system between the robots and the central data processing unit		
Input:	Software flowchart, decisions on software libraries		
Dependencies:	Software architecture design		
Risks:	Wireless hardware is unreliable or interfaces poorly with other software or hardware		

WBS#:	1.2.2.5	Task:	SDP Subsystem
Est. Effort (hrs):	15	Owner:	Rachel
Resources:	SDP research, implementations	Work products:	Complete SDP functions
Description:	Create a flexible scheduling, distribution, and planning subsystem that efficiently assigns work to robots		
Input:	Software flowchart, decisions on software libraries		
Dependencies:	Software architecture design		
Risks:	SDP algorithms are not efficient enough to meet requirements		

WBS#:	1.2.3	Task:	Subsystem Testing
Est. Effort (hrs):	4	Owner:	All
Resources:	Software subsystems	Work products:	Complete software subsystems
Description:	Test all software subsystems to ensure that they give the expected output when provided with testing inputs		
Input:	Completed software subsystems		
Dependencies:	All software subsystem tasks		
Risks:	Software subsystems were implemented incorrectly and do not perform to expectations		

WBS#:	1.3.1	Task:	Software Integration
Est. Effort (hrs):	3	Owner:	All
Resources:	Software subsystems	Work products:	Complete software pipeline
Description:	Test integration of all software components by creating an end to end pipeline consisting of all software subsystems		
Input:	Completed and individually verified software subsystems		
Dependencies:	Subsystem testing		
Risks:	Subsystems cannot integrate with each other		

WBS#:	1.3.2	Task:	Full System Integration
Est. Effort (hrs):	3	Owner:	All
Resources:	S.W. and H.W. subsystems	Work products:	Working robot system
Description:	Complete integration of software components with hardware components		
Input:	Completed and individually verified software and hardware subsystems		
Dependencies:	Software integration		
Risks:	Software and hardware cannot interface with one another, models do not work in in practice		

WBS#:	1.3.3	Task:	Requirements Testing
Est. Effort (hrs):	5	Owner:	All
Resources:	Working robot	Work products:	Complete, working robot system
Description:	Verify the reliability and effectiveness of the robot by conducting our full testing suite		
Input:	Unverified but working robot system		
Dependencies:	Full system integration		
Risks:	Robot fails tests, need to rework some subsystems		

2.2 Schedule

Fig.9 shows our current progress towrads meeting our schedule. At this point, the electromechanical design is slightly behind - the camera rig assembly is still being built, and the second robot has not been constructed yet. The camera rig has been recently updated in design, and parts have been ordered. We do not expect assembly to take long, or hinder progress in integration. The second robot parts have already been ordered. We delayed building it to ensure we could finalize the robot design before building a second one.

Software implementation at this point is complete. All of the major subsystems are functional, and the only additions are to enhance or add additional features. Other software changes are being done for integration, to allow the various subsystems to work with the hardware, or with each other. The software design and implementation has reached usability and is therefore on schedule.

Integration is will underway, which is the majority of the work we have left. Some subsystems have been integrated, and others are actively being worked on. Motion onboard the robot is completed, and there are plans to finish integrating the communication and localization subsystems within the next week.

	Week Number	1	2	3	4	5	6	7	8	9	10	11	12	13
WBS	Task	1/30	2/6	2/13	2/20	2/27	3/6	3/20	3/27	**4/3**	4/10	4/16	4/24	5/1
1.1	Electromechanical													
1.1.1	Finalize CAD Design													
1.1.2	Obtain Parts													
1.1.3.1	Chassis Fabrication													
1.1.3.2	Writing Implement Fabrication													
1.1.4.1	Camera Rig Assembly													
1.1.4.2	Robot Agent Assembly													
1.1.5	Mechanical Testing													
1.2	Software Implementation													
1.2.1	Softare Architecture Design													
1.2.2.1	Localization Subsystem Development													
1.2.2.2	Locomotion Subsystem Development													
1.2.2.3	UI Subsystem Development													
1.2.2.4	Communication Subsystem Development													
1.2.2.5	SDP Subsystem Development													
1.2.3	Software Subsystem Testing													
1.3	Integration													
1.3.1	Software Subsystem Integration													
1.3.2	Full System Integration													
1.3.3	Requirements Testing													
	Demo Preparation													

Figure 9: Schedule via Gantt Chart

3 Requirements Tracking

For readbility we provide a summary of our requirements below.

Requirement	Title
FR1	Omnidirectional Movement
FR2	Autonomous
FR3	Robots Localize Globally and Locally
FR4	Within Bounds
FR5	Insert Writing Tools
FR6	Remove Writing Tools
FR7	Replace Writing Tools
FR8	Coordination
FR9	Drive Control System
FR10	Turn on or off writing tool
FR11	Input Drawing Plan
FR12	Robots Know Progress
FR13	Kill Switch
FR14	User Interface to Robot
NFR1	Docmentation
NFR2	Error Handling
NFR3	Weight Restriction
NFR4	Size Restriction
NFR5	Efficiency
NFR6	Quality
NFR7	Battery Power
NFR8	Reliability
NFR9	Reliable Communication
NFR10	Budget
NFR11	Safe
NFR12	Positional Accuracy
NFR13	Rotation Accuracy
NFR14	Tool Switching Duration

3.1 Objectives Tree

We created an objectives tree to better organize our requirements, and prioritize them based on system purposes and goals. After analysis of our requirements, we formed the objectives tree in Fig.10 with the following categories:

1. Is Safe (Fig.11)
2. Is Portable (Fig.12)
3. Drawing Tool is Easy to Operate (Fig.13)
4. Is Mobile (Fig.14)
5. User-Friendly (Fig.15)
6. Performance Guarantees (Fig.16)

The category for 'Is Safe' (Fig.11) encompasses requirements for the robot staying within bounds, maintaining reliable communication, existence of a kill switch, and overall safe operation. These requirements breakdown how safe usage of the robot can be achieved, through both system design and user operation.

'Is Portable' (Fig.12) specifies system constraints that enable the robots to be able to be transported easily. The battery-powered requirement ensures the robots do not need external power during operation. Weight and size requirements were further categorized into physical constraints, to emphasize the importance of those requirements on portability outside of system operation.

Subtree 'Drawing Tool is Easy to Operate' (Fig.13) ensures the writing tool is easy to maintain and use both during and before or after system operation. The main subtree describes tool maintenance. Requirements under maintenance include inserting, removing, and replacing the tool, as well as duration

requirements for replacing the writing tool. Other requirements in this category relate to having the ability to enage or disengage the writing tool. This requirement is involved with system operation, and ensures the robot can change the tool status so the robots can move regardless of whether it is drawing.

The 'Is Mobile' (Fig.14) tree categorizes mobility requirements and constraints for the robot agents. Both positional and rotational accuracy are categorized under their own Accuracy subtree. Other leaves in this subtree ensure the robot agents have their own drive control systems, can localize, and are able to move autonomously in any direction on a 2D plane.

We also chose to separate out requirements that relate to engaging the user and enable a user-friendly experience. These fall under the 'User-Friendly' subtree (Fig.15). Both documentation and budget requirements were categorized here - these requirements are more likely to be for users interested in adapting or recreating our system. As a result, other requirements were further categorized into a user-interaction subtree. These constraints denote existence of a UI, error handling, and the ability for users to input their drawing plan.

The final categorization, 'Performance Guarantees' (Fig.16) denotes overall system requirements to ensure the final drawing meets specifications. These requirements include ensuring the robots know their own progress, and coordinate with each other. In addition, requirements specifying system efficiency, reliability, and overall drawing quality fell into this category.

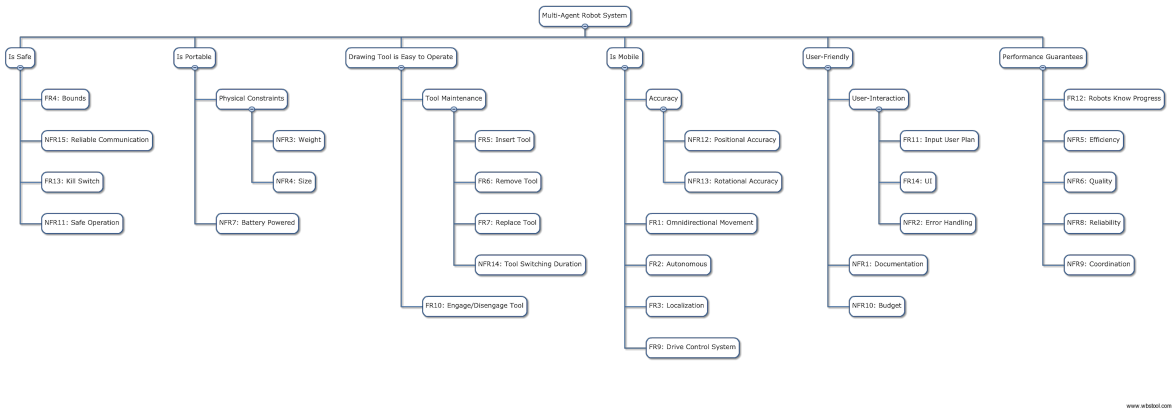

Figure 10: Full Objectives Tree

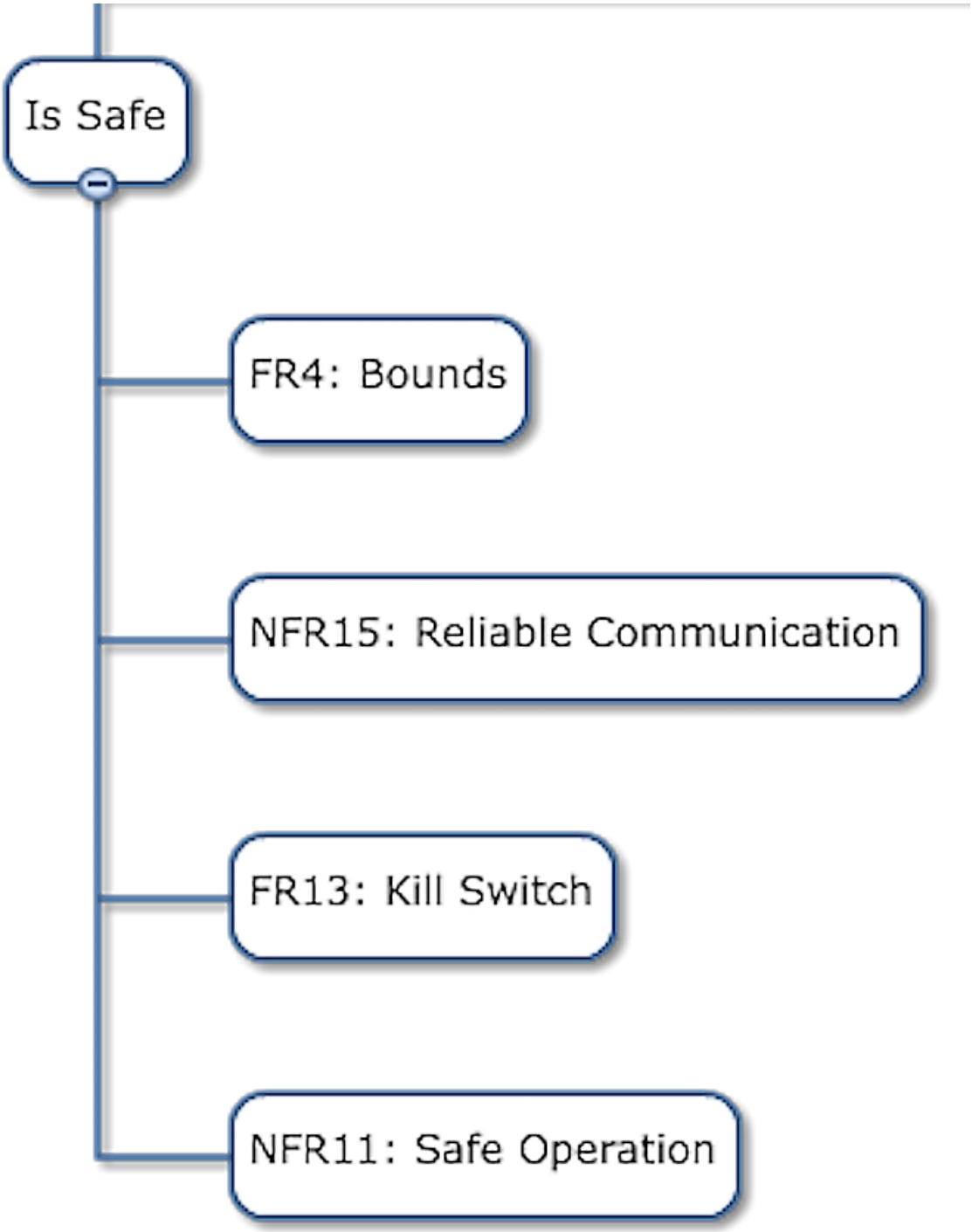

Figure 11: Objectives Tree: Is Safe Branch

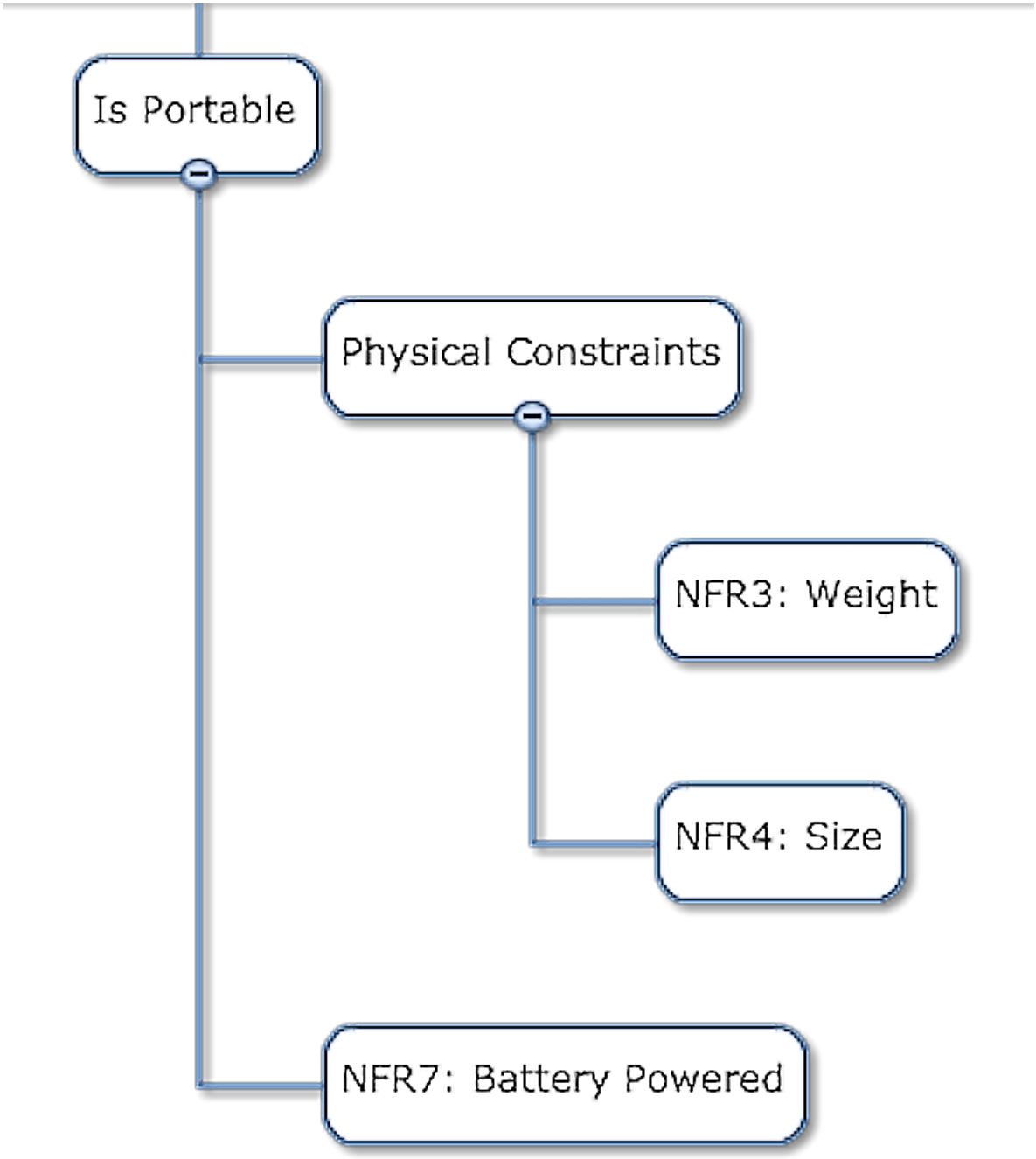

Figure 12: Objectives Tree: Is Portable Branch

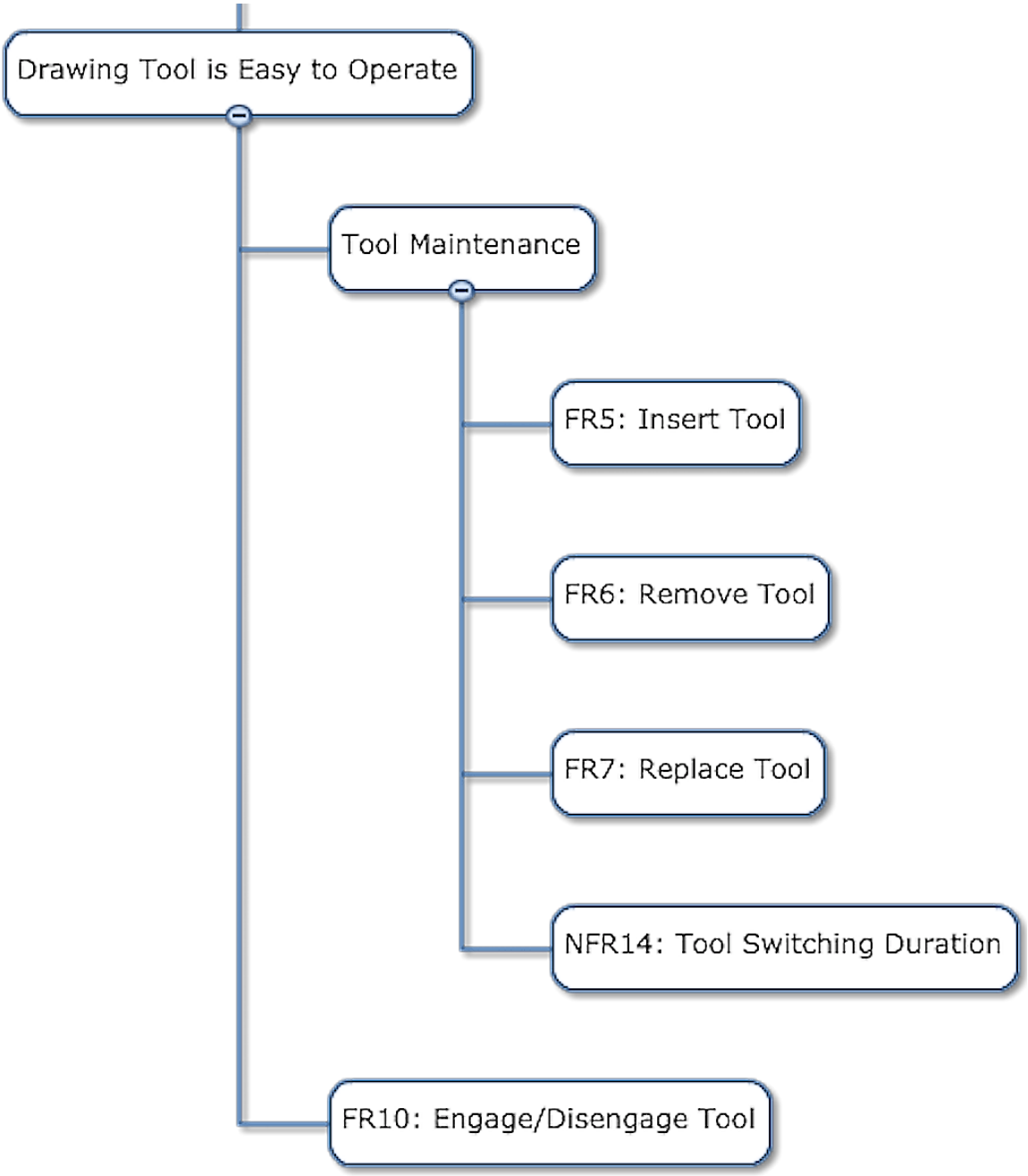

Figure 13: Objectives Tree: Drawing Tool is Easy to Operate Branch

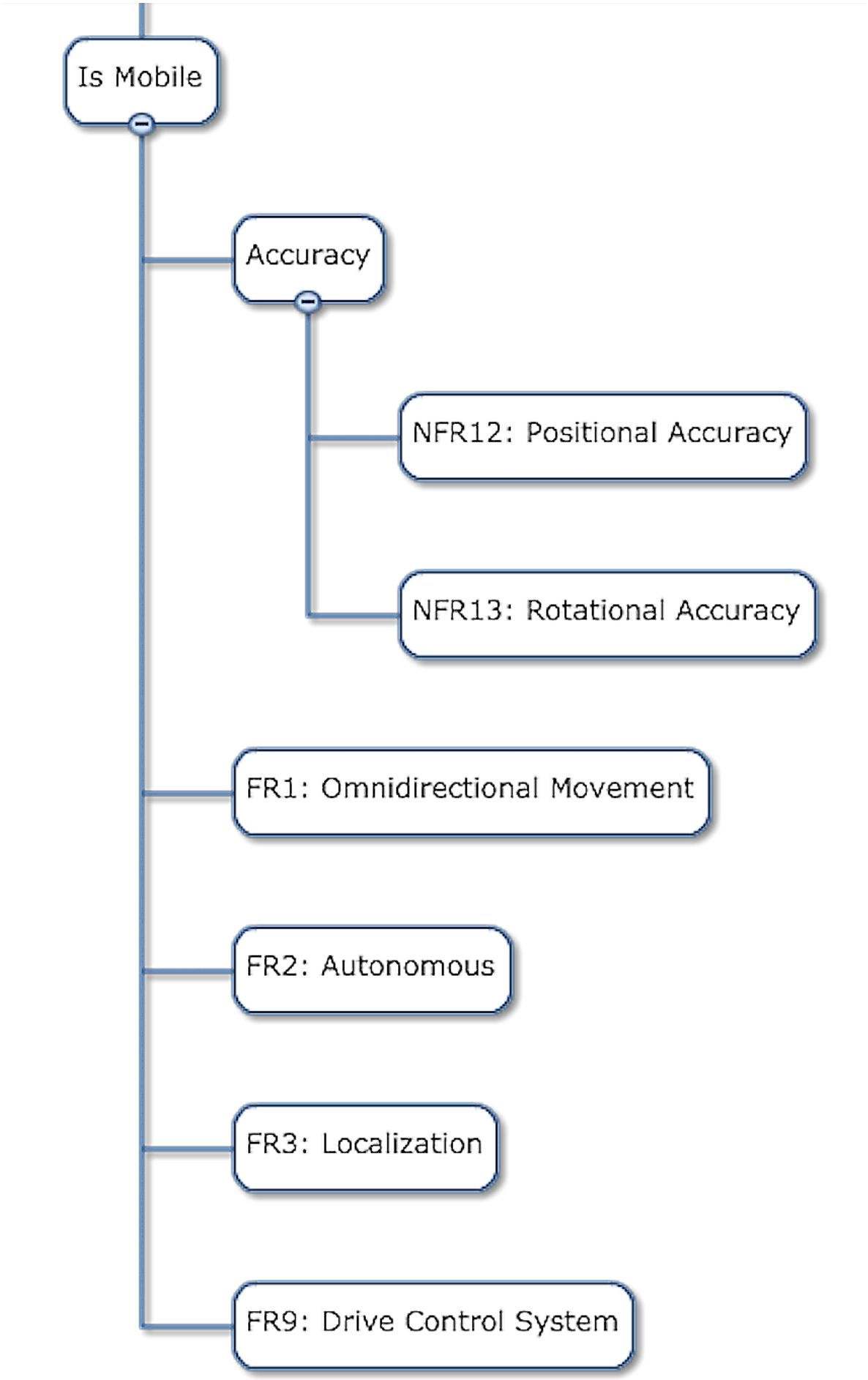

Figure 14: Objectives Tree; Is Mobile Branch

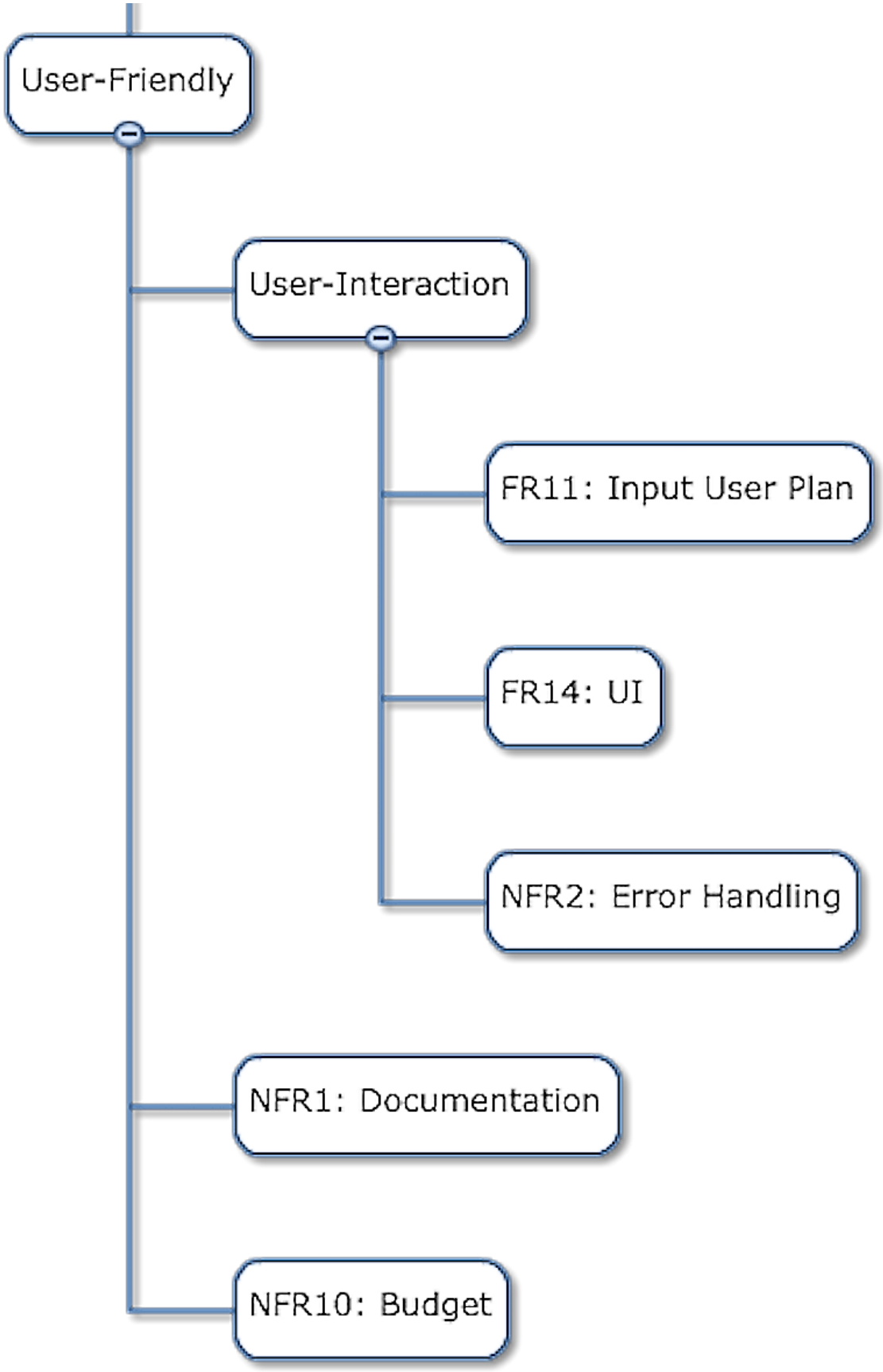

Figure 15: Objectives Tree: User-Friendly Branch

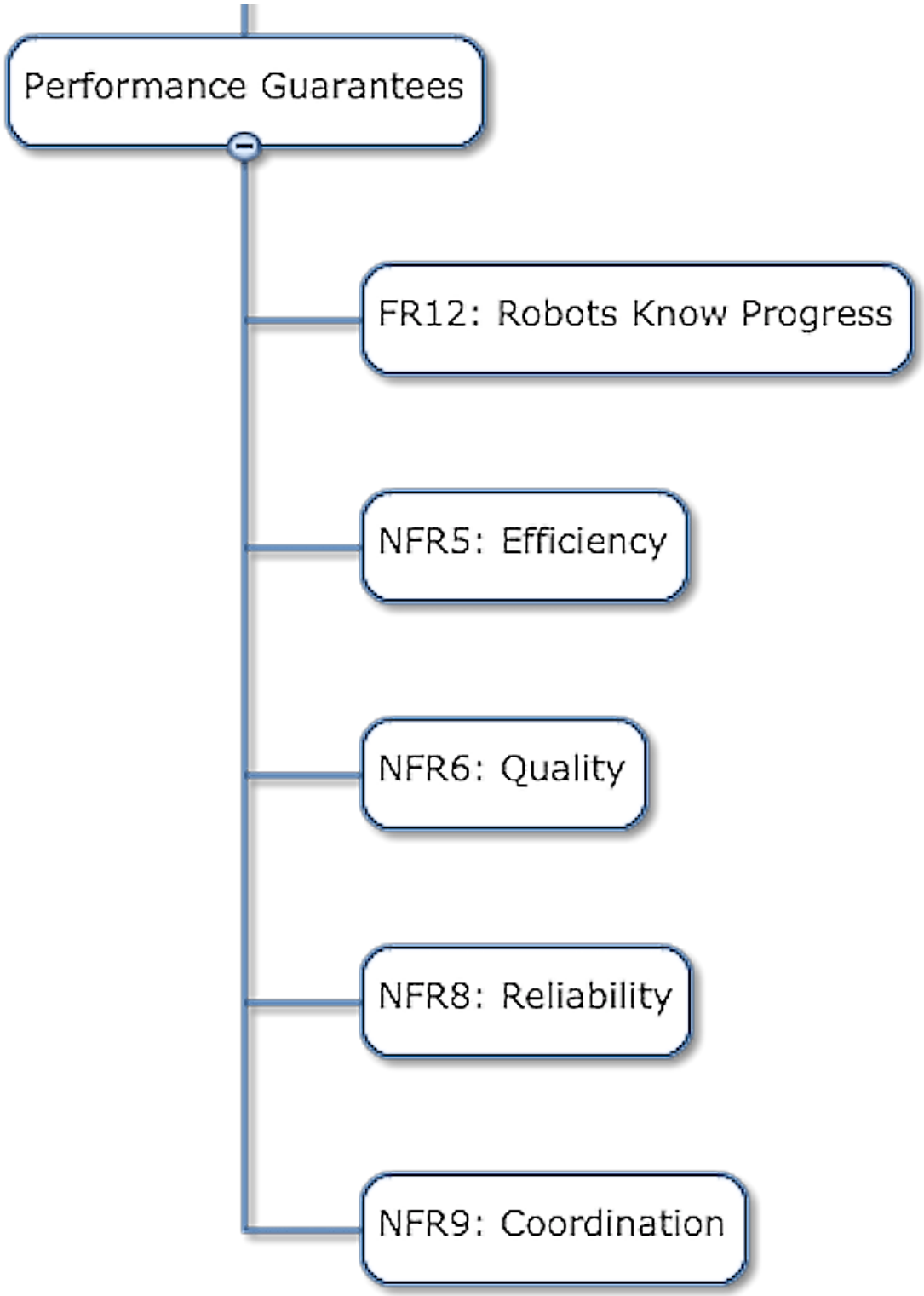

Figure 16: Objectives Tree: Performance Guarantees Branch

3.2 Requirements Traceability Matrix

We present our requirements traceability matrix in Fig.17. We categorize our requirements into functional and nonfunctional requirements. For each requirement, we classify which subsystem it relates to: writing, communication, locomotion, localization, SDP (scheduling, distribution and planning) and mechanical structure. Each requirement is colored green if initial testing has shown that we achieve the requirement.

	Subsystems	Writing	Communication	Locomotion	Localization	SDP	Mechanical Structure
Functional Requirements	FR1: Omnidirectional Motion			X			
	FR2: Autonomous			X	X	X	
	FR3: Localize Globally and Locally				X		
	FR4: Within Bounds			X	X		
	FR5: Insert Tool	X					
	FR6: Remove Tool	X					
	FR7: Replace Tool	X					
	FR8: Coordination					X	
	FR9: Drive Control			X			
	FR10: Turn on/off Tool	X					
	FR11: Input Plan		X			X	
	FR12: Know Progress		X		X	X	
	FR13: Kill Switch	X	X	X			
	FR14: User Interface		X			X	
Nonfunctional Requirements	NFR1: Documentation	X	X	X	X	X	X
	NFR2: Error Handling	X	X	X	X	X	X
	NFR3: Weight Restriction						X
	NFR4: Size Restriction						X
	NFR5: Efficiency					X	
	NFR6: Quality	X		X	X		
	NFR7: Battery Power	X		X			
	NFR8: Reliability	X	X	X	X	X	X
	NFR9: Reliable Communication		X				
	NFR10: Budget						X
	NFR11: Safe			X	X	X	X
	NFR12: Positional Accuracy			X	X		
	NFR13: Rotational Accuracy			X	X		
	NFR14: Tool Switching Duration	X					X

Figure 17: Requirements Traceability Matrix

4 Risk Management

In this section, we revisit the risks defined by our previous document. The risk tables have been updated with the actions we've taken to address the risks.

Risk ID:	Risk Title:	Risk Owner:	Actions taken:
1	Defective Parts	Don	
Description:			
Parts that we ordered arrived defective or do not perform to specifications			No issues Persistent: Only ordering reliable parts
Consequences:	Risk Type:		
We need to reorder parts, expending extra time and budget	Parts		
Risk Reduction Plan:	Expected Outcome		
We will order only parts that have been extensively reviewed, or we have experience with, and order extra parts	We will be able to properly deal with any parts that break during the development process		

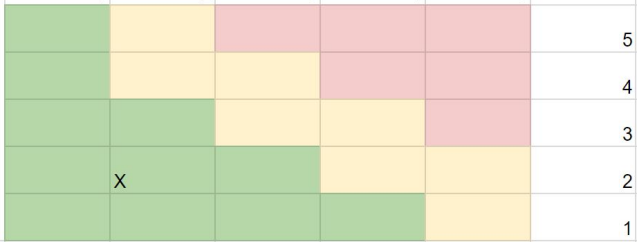

Figure 18: Risk 1: Defective Parts

Risk ID:	Risk Title:	Risk Owner:	Actions taken:
2	Unvavailable Group Member	All	
Description:			
A group member becomes unavailable for work due to travel, sickness, or other emergencies			No issues Persistent: Only ordering reliable parts
Consequences:	Risk Type:		
Work that would have been distributed to that group member needs to be reassigned	Logistical		
Risk Reduction Plan:	Expected Outcome		
We will ensure that every group member is always on the same page about progress so we don't lose too much progress	If a member becomes unavailable, it will only be for a short time and can be easily dealt with		

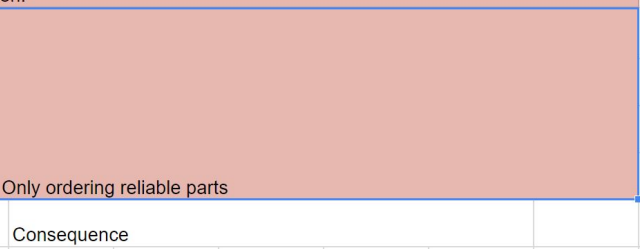

Figure 19: Risk 2: Unavailable Member

Risk ID:	Risk Title:	Risk Owner:	Actions taken:
3	Breaking parts	Eric	3/28/17: Motor shaft broken, reprinted 3/30/17: Motor encoder magnet flawed, needed realignment Persistent: Careful handling of parts
Description:			
Parts unexpectedly break as a result of accidents or improper use			

Consequences:	Risk Type:		Consequence					
We need to reorder parts, expending extra time and budget	Parts		1	2	3	4	5	
								5
Risk Reduction Plan:	Expected Outcome							4
								3
We will practice safe procedures when working with parts and order extras in case	Few parts will break, and even if they do we will have extras on hand	Likelihood				X		2
								1

Figure 20: Risk 3: Breaking Parts

Risk ID:	Risk Title:	Risk Owner:	Actions taken:
4	Mecanum Drive Too Unstable	Eric	4/1/17: Tested mecanum wheels, no issues present Persistent: Built extra time into schedule
Description:			
The drive mechanism for the robot proves too be too unstable or unreliable for our purposes			

Consequences:	Risk Type:		Consequence					
We will need to redesign the drive mechanism, expending considerable time and effort	Design flaw		1	2	3	4	5	
								5
Risk Reduction Plan:	Expected Outcome							4
						X		3
We will build enough time in our schedule to deal with it if necessary, and will use suspension	The instability resulting from the wheels will be manageable	Likelihood						2
								1

Figure 21: Risk 4: Mecanum Drive Too Unstable

Risk ID:	Risk Title:	Risk Owner:	Actions taken:
5	Localization not precise enough	Neil	4/8/17: Preliminary test of localization, appears robust enough Persistent: None
Description:			
Our localization system is not precise enough to ensure that the drawings are accurate representations of input			

Consequences:	Risk Type:		Consequence					
We will need to redesign the localization system or redefine drawing requirements	Design flaw		1	2	3	4	5	
Risk Reduction Plan:	Expected Outcome							5
We will test the localization system early on in order to catch any design flaws within the system	Localization will work well enough for our purposes							4
								3
					X			2
		Likelihood						1

Figure 22: Risk 5: Localization not precise enough

Risk ID:	Risk Title:	Risk Owner:	Actions taken:
6	Unexpected Budget Overruns	Rachel	No issues Persistent: We have a significant buffer in our budget
Description:			
We unexpectedly run out of budget, because parts cost more than expected or other parties reduce our budget			

Consequences:	Risk Type:		Consequence					
We need to scale down our project, or possibly even acquire funds through other means	Logicstical		1	2	3	4	5	
Risk Reduction Plan:	Expected Outcome							5
We will leave a significant buffer in our budget in case unexpected situations occur	We will have a large enough buffer that essential components will be acquired							4
								3
				X				2
		Likelihood						1

Figure 23: Risk 6: Unexpected Budget Overruns

5 Testing and Evaluation Plan

5.1 Design Verification

This section details our test verification plan. This includes the questions and tests we intend to address, and how we plan to measure and achieve success with respect to these tests.

5.2 Writing Implement

5.2.1 Performance Test: Loading

Test Question: Is a human operator able to reload the writing tool within the required time limit of 10s, how long does it take?
Operational Procedure: With a writing tool already installed in the writing assembly, a human test subject will perform 3 reload tests which are separately timed.
Metric: Average duration of the reload time.
Acceptance Criteria: The average reload time is under 10s.
Requirement(s) Verified: NFR14, FR5, FR6, FR7

5.2.2 Performance Test: Writing Quality

Test Question: Is the drawing produced by the robot of acceptable quality?
Operational Procedure: Using a fully loaded writing tool, the robot attempts to draw along a route with at least 4 ft. of travel distance and 3 turns exceeding 50 degrees. Verify that the resulting drawing is of acceptable quality.
Metric: Percent thickness of the route at its narrowest point compared to the maximum thickness of a line created with the writing tool. Boolean on whether or not there are complete breaks in the line.
Acceptance Criteria: The percent thickness is at least 70%, and there are no complete breaks in the line.
Requirement(s) Verified: NFR6

5.2.3 Functional Test: Simultaneous Driving and Writing

Test Question: Can the writing tool make a mark while driving?
Operational Procedure: With a fully loaded writing implement, the robot will mark a 1 ft. line on the writing surface.
Metric: Whether or not any discernible mark is made and the full distance is covered.
Acceptance Criteria: A discernible mark must be made and the full distance must be travelled without the robot becoming stuck or breaking.
Requirement(s) Verified: NFR6

5.2.4 Functional Test: Marking

Test Question: Does the writing tool make a mark when pushed down?
Operational Procedure: The robot will press down on a writing surface with a fully loaded writing implement. Robot must mark with pressure necessary to mark the surface without damaging the surface or writing tool.
Metric: Whether or not a any discerible mark is made.
Acceptance Criteria: A discernible mark must be made.
Requirement(s) Verified: NFR6

5.3 Locomotion

5.3.1 Performance Test: Accuracy

Test Question: Is the robot able to drive with positional and rotational accuracy?
Operational Procedure: The robot drives along a predetermined testing route consisting of driving forward at least 3 feet and driving to its left at least 1 feet.
Metric: The difference of the robot's final position and orientation from the intended position and orientation.
Acceptance Criteria: During the forward driving test, the robot's position must be less than 1 inch

away from the intended position. During the left driving test, its orientation must be less 10 degrees from the commended perpendicular line. This result must be achieved 90 percent of the time.
Requirement(s) Verified: NFR12, NFR13

5.3.2 Functional Test: Speed

Test Question: Can the robot reach a desired speed?
Operational Procedure: The robot will drive along a straight line for 5 ft. during a timed trial.
Metric: The time required for the robot to reach the end of the line.
Acceptance Criteria: The robot must reach the end of the 5 ft. testing course in 20 seconds. This test must be repeatable 90 percent of the time.
Requirement(s) Verified: NFR5

5.3.3 Functional Test: Omnidirectional

Test Question: Can the robot drive omnidirectionally?
Operational Procedure: The robot will drive along a rectangular path that is 10 inches by 10 inches.
Metric: Whether or not the robot can complete the course with only linear motion.
Acceptance Criteria: The robot must be able to complete the course with linear motion only. **Requirement(s) Verified:** FR1, FR9

5.4 Localization

5.4.1 Performance Test: Robot Position Accuracy

Test Question: Is the localization system able to accurately determine the position of each robot?
Operational Procedure: Both robots sit stationary within the working bounds. The localization system then attempts to determine their locations.
Metric: The difference of the robot's actual position from the position reported by the localization system.
Acceptance Criteria: The reported position must be within 1/10 in. of the actual position.
Requirement(s) Verified: NF6, FR4

5.4.2 Performance Test: Bounds Accuracy

Test Question: Is the localization system able to accurately determine the boundaries of the workspace?
Operational Procedure: The localization system attempts to determine the bounds of the workspace.
Metric: The total difference in distance between the reported corners of the workspace and the distance between the actual corners.
Acceptance Criteria: The total difference must not exceed 1 in.
Requirement(s) Verified: FR4

5.4.3 Functional Test: Robot Position

Test Question: Can the localization system find the robot?
Operational Procedure: With a single robot within the working bounds, the localization system attempts to determine the robot's location. Robot operation out of bounds is also considered out of scope, and is undefined behavior.
Metric: Whether or not a location is returned by the localization system.
Acceptance Criteria: The system must return a location for the robot.
Requirement(s) Verified: FR4, NFR6, FR3, NFR13

5.4.4 Functional Test: Bounds

Test Question: Can the localization system find the working bounds?
Operational Procedure: The localization system attempts to find all four corners of the working bounds while they are all in its field of view.
Metric: Whether or not locations are returned for all four corners.
Acceptance Criteria: The system must return locations for all four corners of the working bounds.
Requirement(s) Verified: FR4, NFR6, FR3

5.5 Input Processing

5.5.1 Functional Test: Return Data

Test Question: Does the image processor return data usable by the planner?
Operational Procedure: A user inputs valid drawings via the UI and the system attempts to generate motion commands.
Metric: Whether or not usable motion output is produced.
Acceptance Criteria: The system must be able to produce a series of legal motion commands.
Requirement(s) Verified: FR11

5.5.2 Functional Test: Reject Improper Input

Test Question: Is the input processor able to detect and reject improper input from the UI?
Operational Procedure: The system takes in an invalid input. For example, lines on top of each other.
Metric: Whether or not input is rejected.
Acceptance Criteria: The invalid input must be rejected.
Requirement(s) Verified: FR4, FR11

5.6 Work Scheduling, Distribution and Planning

5.6.1 Performance Test: Executable Plans

Test Question: How consistent is the planner at generating executable plans, ie those that avoid collision and stay within bounds?
Operational Procedure: Using the example input set (Appendix A), run each input and check the plan for potential robot-robot collisions and out of bounds driving.
Metric: Ratio of number of unacceptable plans, those that would involve collision or driving out of bounds, over the total number of plans.
Acceptance Criteria: Almost all, 99% of plans would not involve collision or out-of-bounds if executed.
Requirement(s) Verified: FR4, NFR11

5.6.2 Performance Test: Execution Distribution

Test Question: How efficiently is execution time, i.e. the total time robots spend moving, distributed?
Operational Procedure: Using the example input set (Appendix A), run each input and record the total time each robot spends moving.
Metric: We define execution efficiency as $\frac{\min(execution(R_0),execution(R_1))}{\max(execution(R_0),execution(R_1))}$ where execution(R_0) refers to the execution time of robot 0 and execution(R_0) refers to the execution time of robot 1
Acceptance Criteria: Execution efficiency of 0.75.
Requirement(s) Verified: FR8, NFR5

5.6.3 Performance Test: Drawing Distribution

Test Question: How efficiently is drawing time, i.e. the total time robots spend drawing, distributed?
Operational Procedure: Using the example input set (Appendix A), run each input and record the total time each robot spends drawing.
Metric: We define drawing efficiency as $\frac{\min(draw(R_0),draw(R_1))}{\max(draw(R_0),draw(R_1))}$ where draw(R_0) refers to the drawing time of robot 0 and draw(R_0) refers to the drawing time of robot 1
Acceptance Criteria: Drawing efficiency of 0.75.
Requirement(s) Verified: FR8, NFR5

5.6.4 Performance Test: Speedup

Test Question: What speedup is achieved by using two robots instead of one?
Operational Procedure: Using the example input set (Appendix A), run each input first with one robot and then with two. Time the execution time of each variant.
Metric: The comparison of duration, i.e. $\frac{executiontimewith2robots}{executiontimewith1robot}$.
Acceptance Criteria: According to our requirements we expect a speedup of 2x.
Requirement(s) Verified: NFR5

5.6.5 Functional Test: Collision Free

Test Question: Does the planner and executor generate collision free plans?
Operational Procedure: Using the example input set (Appendix A), run each input and check for any robot-robot collisions during execution.
Metric: Boolean across each plans on whether a collision occurred.
Acceptance Criteria: We only accept if collisions were avoided on 95% of our test cases.
Requirement(s) Verified: NFR 11

5.6.6 Functional Test: Autonomy

Test Question: Does the system require no user input beyond adding the image to be drawn (except for error handling)?
Operational Procedure: After having input a plan, press "Run" on the system and observe if the system requires user input to finish the drawing.
Metric: Boolean on whether user input was required, excluding input relating to errors.
Acceptance Criteria: Accept only if no input was required.
Requirement(s) Verified: FR 2

5.7 Communication

5.7.1 Performance Test: Uptime

Test Question: What is the uptime on our ability to communicate data between the robots and the offboard system?
Operational Procedure: Run the system for a significant period of time (several hours) and record any communication downtime or data loss during communication.
Metric: Time duration of down communication and packet loss.
Acceptance Criteria: Operational 95% of the time.
Requirement(s) Verified: FR12, NFR8, NFR9

5.7.2 Functional Test: Sending and Receiving Data

Test Question: Can the robot send and receive data to and from the offboard device and can the offboard device send and receive data to and from the robot?
Operational Procedure: Send data from the off-board device to the robot and verify the robot received it. Send data from the robot to the off-board device and verify the off-board device received it.
Metric: Four booleans on whether the data is successfully sent and recieved on both ends.
Acceptance Criteria: We must succeed on all four accounts.
Requirement(s) Verified: NFR9

5.7.3 Functional Test: Data Parsing

Test Question: Can the data on each side (robot, offboard device) be parsed by each system?
Operational Procedure: Send data from the offboard device to the robot and verify the robot received it and can execute and process the appropriate information. Send data from the robot to the offboard system and verify the offboard device received and can respond to it.
Metric: Check whether the data was successfully parsed on all sides.
Acceptance Criteria: We require all data be parsable.
Requirement(s) Verified: NFR9

5.8 User Interface

5.8.1 Performance Test: Emergency Stop Speed

Test Question: How fast does the emergency stop shut down the system?
Operational Procedure: While the system is in use, press the emergency stop button and time how long it takes for everything to completely shut down.
Metric: Elapsed time.
Acceptance Criteria: It is vital to safety that our emergency stop shuts everything down within a

second.
Requirement(s) Verified: FR13, NFR11

5.8.2 Performance Test: Error Reporting Delay

Test Question: What is the delay between an error occurring and that error being reported to the user?
Operational Procedure: Given a list of known operational errors, intentionally trigger each error within the system and report the time between causing the error and it being reported to the user.
Metric: Averaged elapsed time across error reporting.
Acceptance Criteria: The average time to detect and report an error should be within 3 seconds.
Requirement(s) Verified: NFR2, NFR11

5.8.3 Performance Test: Error Understandability

Test Question: How understandable and informative are error messages?
Operational Procedure: Given a list of known operational errors, intentionally trigger each error while a non-developer user is using the system (while masking the error cause) and evaluate how well the user can determine the error. For example, while the system is drawing, the user could be in a different room with only the error reporting device, making the user unable to see what errors the robots are facing.
Metric: Determine if the user can determine the error and knows how to react to or correct the error.
Acceptance Criteria: The user should be able to determine and react effectively for 90% of the errors.
Requirement(s) Verified: NFR1, NFR2, FR14

5.8.4 Functional Test: Emergency Stop

Test Question: Does the emergency stop fully stop the system?
Operational Procedure: While the system is in use, press the emergency stop button and check if all systems halt their operation.
Metric: Boolean on whether every subsystem stops or not.
Acceptance Criteria: It is only successful if the boolean metric is true.
Requirement(s) Verified: FR13

5.8.5 Functional Test: Error Reporting

Test Question: Is each operational error reported to the user?
Operational Procedure: Given a list of known operational errors, intentionally trigger each error within the system and report whether the error caused it reported to the user.
Metric: Each error must be reported correctly. Hence we can divide the number of correctly reported errors by the number of total errors caused to determine an error-reporting score.
Acceptance Criteria: Considering error handling is critical to performance, our system should have an error-reporting score of 90%.
Requirement(s) Verified: NFR1, NFR2, FR14

5.9 Power System

5.9.1 Performance Test: Battery Life

Test Question: How long can an individual robot run for on a single battery charge?
Operational Procedure: Charge a robot fully. Command the robot to complete an input drawing repeatedly until the robot is fully drained of power. Time how long this takes.
Metric: The duration of operational time given one charge
Acceptance Criteria: We accept this if the operational time exceeds the necessary duration time of 90% of our test drawing inputs.
Requirement(s) Verified: NFR7

5.10 Full System Validation

6 Full System Validation

6.1 Performance Test: Painting Accuracy

Test Question: How closely does the drawn image resemble the original input?
Operational Procedure: Using example input sets for the system to complete. After completion, overlap the input with the image of final drawing captured from overhead camera. Rescale the two images so that they are in the same size. Evaluate the coherence of the two images.
Metric: The percentage of drawn lines that were within 3 pixels of difference compared to those of the original image.
Acceptance Criteria: The system must successfully and accurately draw 95% of the lines in the original input.
Requirement(s) Verified: NFR6

6.2 Performance Test: Reliability

Test Question: How reliable is the system in terms of successfully complete a series of drawing tasks?
Operational Procedure: Command the system to finish a series of drawing tasks. Measure the number of consecutive successful completion. Successful completion is defined as the system autonomously finishes painting and the painting process is free of errors including but not limited to localization breakdown, motor breakdown, or painting mechanism breakdown. Calling human interference with switching battery and drawing utility does not count as unsuccessful run.
Metric: Number of consecutive painting completion.
Acceptance Criteria: The minimum acceptable number of consecutive completion is 5.
Requirement(s) Verified: NFR8

6.3 Functional Test: Size

Test Question: Is the robot agent too big to be portable, i.e. carry the robot through a standard door?
Operational Procedure: Measure the physical dimensions of the robot in terms of width, length, and height or in terms of diameter and height.
Metric: Numeric value of each length measurement; robot footprint; robot volume.
Acceptance Criteria: Must be less than 80 in. x 36 in. x 36 in.
Requirement(s) Verified: NFR4

6.4 Functional Test: Weight

Test Question: Is the robot agent too heavy to be portable, i.e. able to be lifted by a normal person?
Operational Procedure: Measure the mass of the robot.
Metric: Numeric value of robot mass.
Acceptance Criteria: Must be less than 50 pounds.
Requirement(s) Verified: NFR3

6.5 Functional Test: Budget

Test Question: Does the cost for developing this robotic system exceed our budget?
Operational Procedure: Document total amount of money spent for designing and constructing this robot system. This includes machining expense, part cost, and etc.
Metric: Total amount of money spent.
Acceptance Criteria: Total developing expense has to be less than $2500.
Requirement(s) Verified: NFR10

6.6 Functional Test: Safety

Test Question: Is the robot safe during operation? Specifically, when collision happens, will the robot harm other robots, external environment, or human?

Operational Procedure: Count the number of sharp edges on the exterior of the robot. Also, measure the time it takes from the overhead camera detects collision to robot agent stops moving motors. Intermediate steps involved are: camera sends collision signal to system controller and system controller commands involved robot agent to stop its current action.
Metric: Number of sharp edges (angles less than 90 degrees); amount of time takes from detection to action.
Acceptance Criteria: Values for these two metrics need to be as small as possible. Zero sharp edges can be on the exterior - any edges from, for example, a rectangular chassis, should be rounded. The maximum amount of time is 1.5 seconds.
Requirement(s) Verified: NFR11

6.7 Functional Test: Documentation

Test Question: Is the documentation for the developing process comprehensive and replicable?
Operational Procedure: Give the full documentation to another design group or stakeholder and inquiry if they can duplicate the project with those documents.
Metric: Boolean on whether or not reviewers can replicate the system development process.
Acceptance Criteria: Reviewers are confident to replicate system development process based on the documentation.
Requirement(s) Verified: NFR1

Appendices

A Planner Inputs

The following are a set of sample drawing inputs that will be used to test the system. Some test inputs have been randomly generated while others were designed to stress test a particular feature.

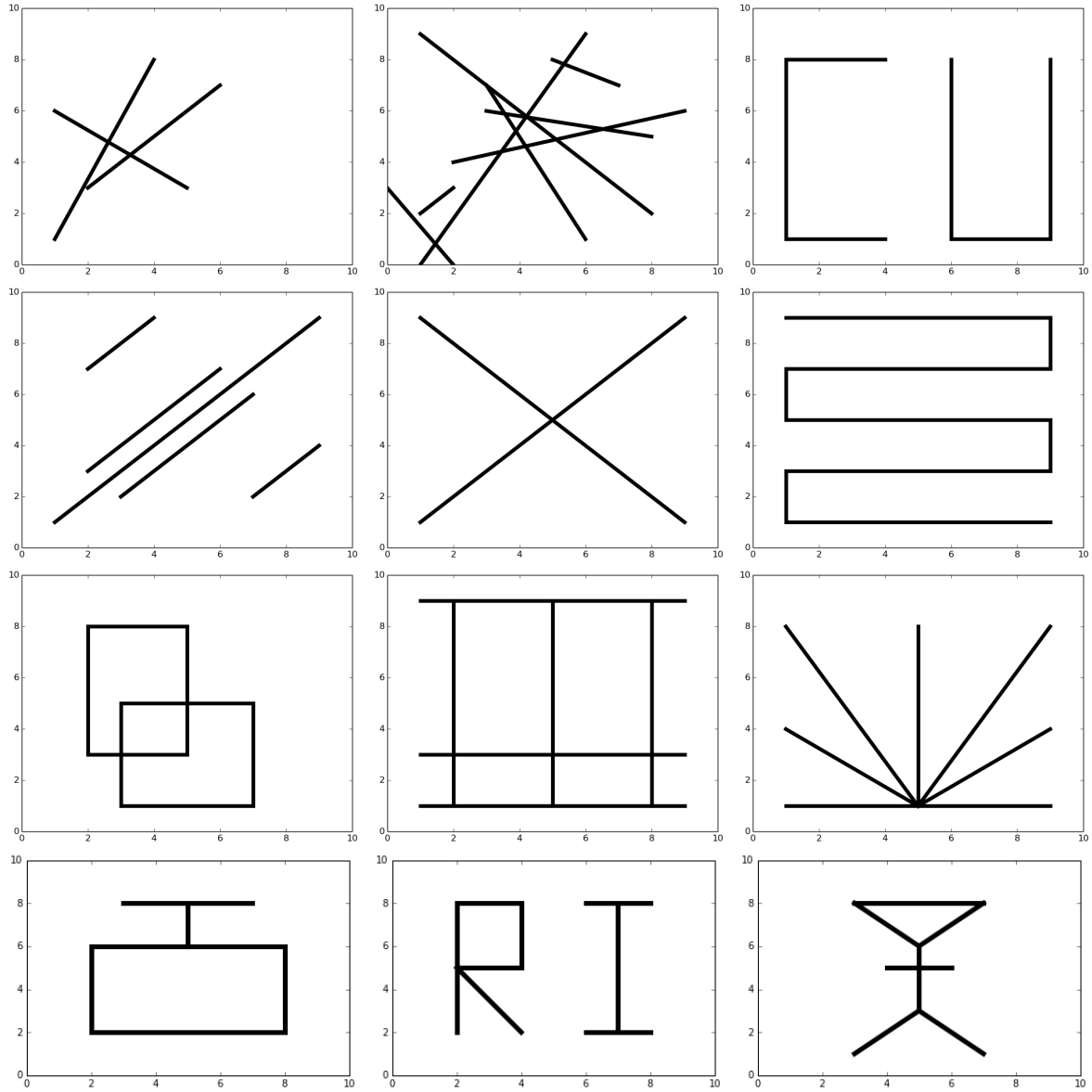

Figure 24: Planner Test Cases.