SpaceOps-2025, ID # 30

Work Models that Compute: Informing Function Allocation in Mission Operations Systems through Modeling and Simulation of Cognitive Agents

Mason Starr[a]*, Glenn Lightsey[b], Karen Feigh[c]

[a] *College of Aerospace Engineering, Georgia Institute of Technology, United States of America*, mstarr32@gatech.edu
[b] *College of Aerospace Engineering, Georgia Institute of Technology, United States of America*, glenn.lightsey@gatech.edu
[c] *College of Aerospace Engineering, Georgia Institute of Technology, United States of America*, karen.feigh@gatech.edu

* Corresponding Author

Abstract

All space missions, from crewed space stations to mega-constellations and highly autonomous surface vehicles, require an integrated network of humans and technology to function. Work Models that Compute (WMC) provides a modeling and simulation framework built for early design of ConOps for complex multi-agent socio-technological systems. Built on the tenets of Work Domain Analysis, WMC is ideal for rapid prototyping of function allocations, authority structures, and teamwork configurations within a mission operations environment. WMC models work separately from agents, simulating agent-to-agent interactions within the work environment as agents work towards mission objectives. The interplay between system dynamics, agents, their environment, and work function allocation reveals an emergent ConOps and operational timelines that can be assessed with performance metrics to inform early trade-space decisions. This work seeks to apply WMC to a real-world example: Rotating Trajectory Correction Maneuvers (RTCMs) performed during the Lunar Flashlight mission. RTCMs were developed as an operational adaptation to mitigate an anomalous propulsion system, requiring precise ground-in-the-loop control to avoid reaction wheel saturation. By implementing the human-spacecraft work of RTCMs within WMC, this work evaluates whether the framework can accurately replicate complex system dynamics and real-world operations and provide insights into potential alternative function allocations. Following validation, WMC simulations will be used to assess hypothetical scenarios where greater autonomy is shifted to the spacecraft, testing whether onboard phase queuing and telemetry-triggered commands could have improved operational efficiency.

Keywords (6): Work Domain Analysis, Modeling and Simulation, ADC, Propulsion, Teleoperation, ConOps.

Nomenclature

$\vec{\tau}$	=	Torque imparted on spacecraft due to thruster firings.
$\vec{h}$	=	Total spacecraft angular momentum, including reaction wheels and body rotations.
θ	=	Spacecraft rotational phase about thrust vector.
t	=	Time, in seconds.

Acronyms/Abbreviations

Abstraction Hierarchy (AH)
Advanced Spacecraft Energetic Non-Toxic (ASCENT) Propellant
Attitude Determination and Control System (ADCS)
Cognitive Work Analysis (CWA)
Concept of Operations (ConOps)
Deep Space Network (DSN)
Foreign Object Debris (FOD)
Function Allocation (FA)
Georgia Institute of Technology (GT)
Ground Data System (GDS)
Ground-in-the-Loop (GitL)
Jet Propulsion Laboratory (JPL)
Lunar Flashlight (LF)
Mission Operations Center (MOC)
Mission Operations System (MOS)

Rotating Trajectory Correction Manuever (RTCM)
Rotations Per Minute (RPM)
Runge-Kutta 4 (RK4)
Semi-autonomous MTAK Momentum Management and Reactive Timing Script (SMARTS)
Work Models that Compute (WMC)

1. Introduction

1.1 Research Objectives

The study of work is not an academic exercise, but a practical necessity. Especially in space mission operations, humans exist in and interact with increasingly complex systems. Well-designed sociotechnical systems can enable unbelievable feats, while dysfunctional systems can doom a mission regardless of technological capability, budget, or effort. Designing effective and resilient work systems that enhance safety, productivity, and health is challenging. Large problem spaces, social complexity, dynamic and hazardous environments, heterogeneous perspectives, increasingly capable automation, and the envisioned world problem all threaten the operational capabilities of a mission. To study, model, and optimize work is to directly invest in the success of technologies that are ultimately bounded by the realities of human behavior.

The primary objective of this work is to demonstrate the applicability of WMC to satellite operations, something that has not yet been shown. This paper presents a preliminary study in which WMC is used to model work that occurred in the Lunar Flashlight MOS while performing RTCMs. Two scenarios are explored: a realistic scenario modeling the RTCM as performed in flight, and a hypothetical scenario modeling full spacecraft autonomy. The resulting ConOps for each scenario are described and discussed.

1.1.1 Paper Structure

In the introduction, a brief overview of Work Domain Analysis (section 1.2) sets the stage for understanding the work performed during LF RTCMs (sections 1.3, 1.4). An overview of WMC (section 1.5) is required to understand the details of model implementation (section 2). Results from the simulation (section 3) are discussed and compared to flight data (4.1), with an emphasis on future work required for a full study (4.2). The project is summarized and concluded (section 5) with lessons learned identified.

1.2 Work Domain Analysis

Work Domain Analysis is the first step of Cognitive Work Analysis: a framework for designing, analyzing, and evaluating complex sociotechnical systems by focusing on the cognitive demands of human work and how those demands interact with the system at large [1]. WDA is a systematic approach for characterizing elements, structures, goals, relationships, and constraints within a work system. Work is modeled independently of the actors performing it; WDA focuses on the *why* and *what* of a work domain, not *how* work is implemented. By analyzing a work environment in this way, WDA provides a model that supports trade decisions and informs system design while remaining unburdened by details of implementation.

The primary product of a WDA is an Abstraction Hierarchy (AH), a stratified hierarchy of work domain elements linked by means-end relationships (Fig. 1). Abstraction hierarchies are typically divided into five distinct tiers of increasing abstraction, presented from highest level of abstraction to lowest:

- Functional purposes define the highest-level goals and purposes of the system.
- Abstract functions detail high level constraints, principals that guide decisions, and measures of performance.
- Generalized functions define the roles, responsibilities, and tasks required to achieve abstract functions.
- Physical functions define the processes and functions that physical forms in the work system enable.
- Physical forms are natural and manufactured objects used in the work system, including spacecraft, personnel, software, tools, information, celestial bodies, etc.

Elements in an AH are linked to one or more elements in upper and lower tiers by means-end relationships in which the lower element is the means to the end of the upper element. For example, situational awareness is a means to the end of risk management. Means-end relationships are orthogonal to the part-whole relationships typically used in engineering. Although means-end relationships are drawn between elements in an AH, these relationships are typically omitted from AH figures to increase readability.

Functional Purposes	Science Goals	Technology Goals	Social Goals	Maintain MOS Safety				
Abstract Functions	Readiness	Risk Management	Workload	Physics	System Constraints			
Generalized Functions	Activity Execution	Navigation	Maintain Spacecraft Safety	Situational Awareness	Mission Planning	Team Management		
Physical Functions	Commanding & Control	Data Mangement	Communication	Simulation, Analysis, & Modeling	Staffing & Scheduling	Activity Development	MOS Optimization	
Physical Objects	Spacecraft	Testbed	Operators& Personnel	Ground Data System	DSN	Telemetry	Command Products	Data Products

Fig. 1: Abstraction Hierarchy of the Lunar Flashlight mission operations work domain [2]. Elements relevant to the RTCM work in this paper are highlighted in green, and abstract tiers are shown in grey on the left.

After the work in a system is fully understood with WDA, system designers can progress to work function allocation, deciding who, or what, will perform each task. The combination of system dynamics, constraints, and function allocation will give rise to an emergent ConOps detailing how the specified work configuration will function.

The work domain of space operations is an ideal but largely unexplored application of CWA. Space operations require complex teaming of humans and technology to accomplish mission goals, in the form of human, teams, spacecraft, spacecraft subsystems, and ground support automation. The space environment introduces unique constraints that are not applicable in most work domains, driven by light-time communication delays, orbital mechanics, viewing geometry, thermal loads, power consumption, communication budgets, and more. For robotic missions in such an environment, human constraints such as workload are often not weighed as highly, as humans are far more adaptable to new environments than spacecraft. However, human constraints are significant factors in the ConOps of a work system and are perfectly captured and modeled with WDA.

1.3 Lunar Flashlight Mission Overview

Lunar Flashlight (Fig. 2) was a 6U CubeSat developed by NASA JPL and funded by NASA's Space Technology Mission Directorate. LF's primary goal was to demonstrate multiple novel technologies, including the Lunar Flashlight Propulsion System (LFPS): an experimental propulsion system utilizing ASCENT monopropellant, a less-toxic alternative to hydrazine. Although LF was classified as a technology demonstration mission, it had a secondary science objective of using a near-infrared laser array and reflectometer to map the distribution of surface water ice in permanently shadowed regions on the lunar south pole, playing a key role in indentifying lunar *in-situ* resources [3]. A tertiary mission objective was to further partnerships between NASA and universities, and so Georgia Tech was contracted as the primary MOC/GDS for the LF mission, making the LF operations team the first fully student-composed team to operate a beyond-Earth mission.

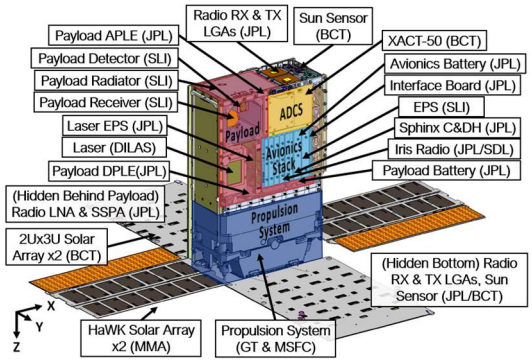

Fig. 2. Lunar Flashlight spacecraft and subsystems.

Lunar Flashlight utilized a ConOps consistent with most beyond-Earth CubeSats. Throughout all mission phases, operators used NASA's Deep Space Network to communicate with spacecraft in two-way contacts, typically 2-3 hours long and 0-3 per day, depending on mission phase. During contacts, operators commanded the spacecraft in real-time, following procedures to build and send commands while monitoring the spacecraft's response. Operators used background command sequences to maintain control over the spacecraft while not in contact.

LF launched on December 11th, 2022. Shortly after launch, operators successfully acquired LF with all systems seemingly nominal. Two days later, the first reaction wheel desaturation burn resulted in an increase of total system momentum, rather than a decrease. This anomalous behavior kicked off a months-long thruster characterization and anomaly recovery campaign. Eventually, the propulsion team traced the likely root cause of the anomaly to FOD in the fuel lines: likely shards of sintered titanium shedding off the additively manufactured fuel manifold [4]. Due to the FOD blockage, the thrust produced by LF's thrusters was highly variable: sometimes a thruster performed well for weeks before dropping thrust completely, sometimes a low-performing thruster would "wake up". On average, the performance of all four thrusters degraded over time.

Unpredictable thrusters posed a significant risk to the mission, as a sudden increase in thrust could saturate the reaction wheels and cause the spacecraft to tumble. The ADCS was programmed for closed loop control of the propulsion system during maneuvers, varying the duty cycle of each thruster during a burn to balance out any unwanted momentum buildup. However, this closed loop control was dependent on predictable thrust. To work around the problem, the LF team developed RTCMs, which allowed the spacecraft to perform extended burns on a single thruster without saturating the reaction wheels. Multiple RTCMs were executed successfully, but unfortunately, FOD continued to accumulate in the fuel lines and eventually no thrust could be achieved. Now on a ballistic trajectory, LF escaped the Earth moon system, leading JPL to declare end of mission in May of 2023.

1.4 Rotating Trajectory Correction Maneuvers

RTCMs are propulsive maneuvers that allow spacecraft to perform extended burns with a single thruster, even if the thruster exerts a large torque on the spacecraft that threatens to saturate reaction wheels. For a complete description of RTCMs, refer to McElrath, Hauge, and Smith [4, 5, 6]. An overview is presented below.

The LF propulsion system consisted of four thrusters mounted on the -Z side of the spacecraft (Fig. 3). Each thruster was positioned near the corners of the spacecraft and canted slightly inward toward the Z-axis. This configuration meant that each thruster, when fired, exerted a significant torque on the spacecraft. Due to their placement far from the spacecraft's center of mass, a single thruster firing at full power could saturate the reaction wheels within seconds.

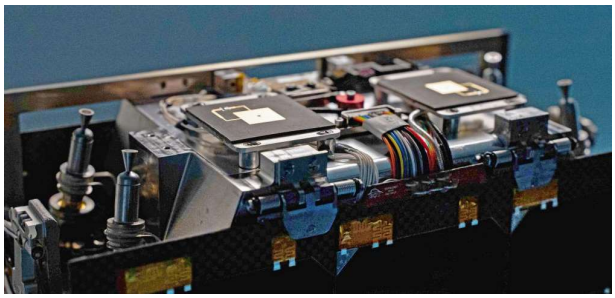

Fig. 3. Lunar Flashlight Propulsion System with four canted thrusters.

When firing a thruster, the spacecraft was subject to a torque $\vec{\tau}$. Within the spacecraft body frame, $\vec{\tau}$ is fixed in direction for a given thruster and scales in magnitude with thruster performance. Note that torque applied over time results in a change to system angular momentum.

During an RTCM, the spacecraft is actively rotated about a single thrust vector by commanding the ADCS to a fixed position in a rotating reference frame. A "burn plane" is defined such that it is normal to the thrust vector and thruster torque is in the $+y$ direction. As the spacecraft rotates, $\vec{h}$ is projected onto the burn plane, tracing a "momentum circle" with $\frac{d}{dt}\vec{h}$ laying tangent to the circle. At a key phase $\theta = 180°$, $\vec{\tau}$ directly opposes $\frac{d}{dt}\vec{h}$. If the thrusters are fired at this key phase, with a specific thrust force magnitude, torque and change in angular momentum will cancel out, preventing angular momentum buildup as the spacecraft continues to burn and rotate. If the thrusters are fired when not at this key phase, the angular momentum will build up and threaten to saturate the reaction wheels. In practice, operators estimated the force of a thruster using DSN Doppler residuals, then performed "setup" burns that adjust the spacecrafts' momentum state such that the magnitude of thruster torque and rate of change of angular

momentum resolved in the rotating frame are equal. Takedown burns were performed after thrusting was complete to lower the spacecraft's momentum state to a safer level.

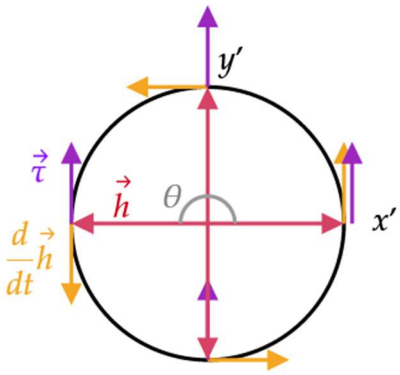

Fig. 4. RTCM momentum circle shown in burn plane. At key phase $\theta = 180°$, thrust torque and change in spacecraft momentum vectors directly oppose each other.

For several weeks, one thruster exhibited consistent and predictable thrust levels, allowing RTCMs to be executed entirely open-loop via pre-written command sequence files, with no ground control in the loop. This approach yielded multiple successful 20-minute-long burns. However, once the single reliable thruster began exhibiting inconsistent behavior, this method became infeasible. Aside from basic fault protection measures to transition to safe mode if the reaction wheel momentum was too high, LF lacked onboard autonomy to adaptively modify maneuver execution in response to thrust variations. The only path to closed-loop thruster control was via human-in-the-loop commanding, in which operators would assess thrust performance and adjust maneuver execution accordingly in real time using a tool called SMARTS.

1.4.1 RTCM Execution using SMARTS

SMARTS was created to enable real-time response to thruster variations during RTCMs, allowing operators to quickly adapt to thruster performance by performing thruster burns, pump reversals, and other propulsion operations with the click of a button. Despite weeks of analysis, app development, procedure writing, and testbed verification, SMARTS was only used twice in flight before the thrusters ceased working entirely. Thus, a WMC simulation of SMARTS usage serves as both validation of the operations that did occur and a prediction tool for the flight maneuvers that never occurred.

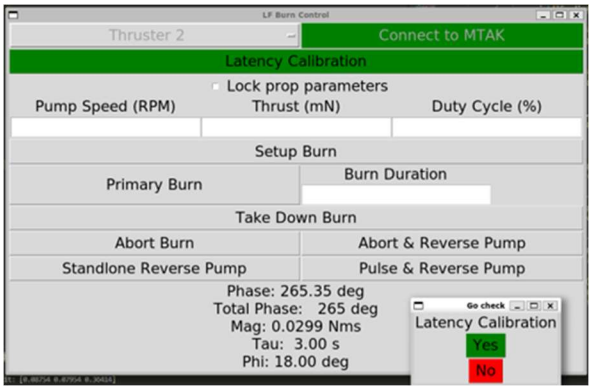

Fig. 5. SMART Graphical User Interface, showing a GO/NO GO popup for executing a latency calibration.

Before using SMARTS, a background command sequence was executed on the spacecraft that initialized the propulsion system, heated the thrusters, slewed to a burn attitude, and initiated a 6° per second (1 RPM) rotation about the thrust vector. As the rotation involved slewing off-sun, power constraints capped the rotation to 20 minutes total. During this time, propulsion operations were commanded from the ground via SMARTS.

The timing of commands during RTCMs had to be precise to properly manage the spacecraft's momentum state. For the torque of a maneuver to oppose the torque of the reaction wheels during a rotation, the thrusters must be fired at a specific rotational phase, to single-second precision. To enable this, a latency calibration was performed to measure

the time delay from telemetry reception on the ground to command execution on the spacecraft. Latency varied due to light-time delay, spacecraft radio usage, and GDS processing times but all factors were accounted for during the calibration. SMARTS internally modeled spacecraft attitude, momentum state, and rotational phase by logging and transforming real-time ADCS telemetry transmitted by LF. Using this internal model and the measured latency, SMARTS reliably queued and dispatched commands for execution at a targeted phase to a precision of ±1°.

Based on thruster performance and momentum state, operators executed command "modules" such as pump reversals to dislodge FOD, setup and takedown burns to adjust spacecraft momentum magnitude, extended primary burns, and burn aborts in the event of bad thrust. SMARTS calculated parameters for and built these commands in real-time, though operators needed to approve any command radiation. Additionally, SMARTS implemented a state machine that enable or disabled certain modules depending on the spacecraft state, to mitigate risk of operators incorrectly sending commands in the wrong order.

When using SMARTS, operators followed a procedural flowchart that guided decision-making in selecting appropriate modules based on spacecraft state, as shown in Fig. 6. Workflow followed by operators when using SMARTS.. Each step in this flowchart represents an action performed by operators, with the assistance of SMARTS for calculations, command queueing, and telemetry processing. The WMC simulation traverses the entire flowchart to perform a complete RTCM. The workflows and processes that the LF operations team followed while using SMARTS are the subject of the WMC simulation.

1.5 Work Models that Compute

WMC is a C++ computational modeling and simulation framework created to support system designers in the development of novel ConOps for complex work systems including human-automation interaction. A principal concept in WMC is that a system's ConOps is not something specified and implemented, but rather the emergent behavior of a system driven by the individuals, environment, actions, and structures that compose it. An overview of WMC is presented below. For a full description of WMC readers are referred to Feigh, Pritchett, and Ijtsma [7, 8, 9].

Work simulators are not a new concept. For example, NASA Ames's Brahms tool has existed since 1998 and has been implemented as an operational aid on missions like Mars Pathfinder. However, WMC is unique in that it models work using the tenets of WDA. Work is modeled separately from the agents performing it, allowing complex work systems to be compiled completely independently of the agents performing the tasks. Work is allocated to agents at runtime, enabling rapid testing of function allocation strategies to quickly inform trade decisions in MOS design, unlike other work modeling tools where task flows are rigid and pre-defined.

WMC was designed with the understanding that work is contextual. Tasks do not occur in isolation but are informed by and act on a larger environment that includes physical, social, and policy-based constraints. WMC relies on multiple constructs to support contextual work modeling (Table 1). *Resources* model the work environment, while *agents* perform *actions* to perceive and affect the environment. The collection of resources, actions, and constraints stemming from an AH that describe a work domain are referred to in WMC as a *work model*. A *scenario* describes the actors present in the simulation and the initial conditions of the environment. A *function allocation* describes how the work is implemented amongst the actors. A WMC simulation requires a work model, a scenario, and an FA to run.

FAs may specify for each action an executing agent that performs the task, and a responsible agent ultimately responsible for the task execution. In this way, WMC can implement authority structures and be used to model human-robot interaction principles such as supervisory control, commanding and confirming actions, and trust. While individual agents perform *taskwork*, WMC can automatically generate and model *teamwork* actions, in which an executing agent and responsible agent coordinate on a single action. Teamwork actions can take the form of an agent monitoring another during task execution, an agent taking direct control of another during execution, an agent commanding another prior to execution, and/or an agent confirming an actions status after completion. FAs specify teamwork configurations, and WMC can automatically generate teamwork actions at runtime.

Running a WMC simulation generates a mission timeline detailing the time and duration of the actions performed by agents. Timelines can be assessed for quantitative metrics that assess system efficiency, including agent workload, agent wait time, system robustness, and failure propagation patterns [10]. These outputs enable mission planners to evaluate how well a given ConOps supports operational performance.

In this study, WMC will be used to assess how RTCM performance would change under different function allocations. Two allocations are considered: one that mimics the allocation used in flight, and one that allocates all functions to the LF spacecraft (full autonomy).

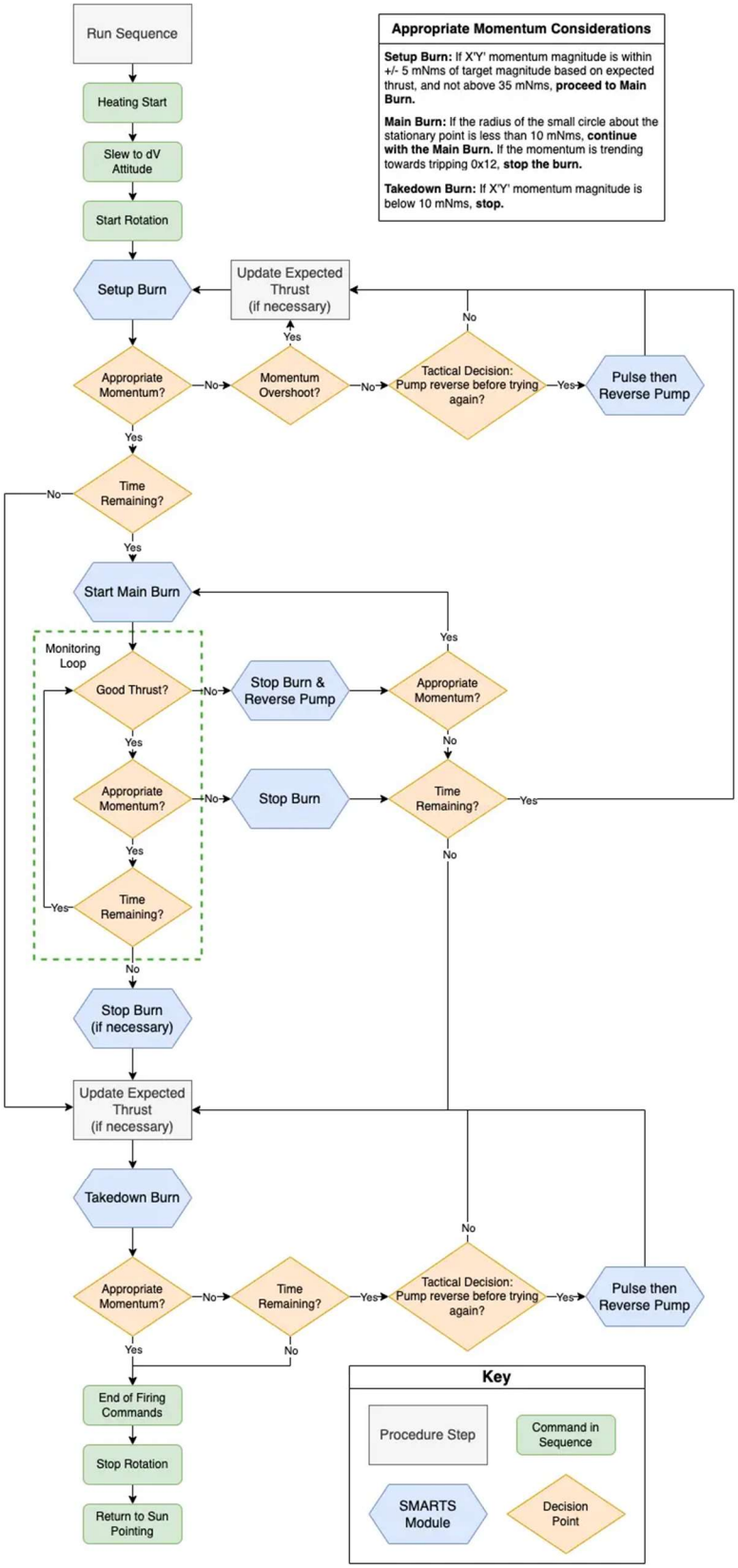

Fig. 6. Workflow followed by operators when using SMARTS.

Table 1. Important WMC constructs. Constructs above the double line are part of a work model, those below are part of a scenario.

Construct	Description	Examples
Information Resource	Variables representing an aspect of the work environment.	Satellite quaternion; location of a physical resource; work strategy.
Physical Resource	Objects required to perform work functions.	Repair tools; ground stations.
Temporal Action	Sample and affect the environment by getting and setting resources. Can be scheduled by other actions or itself.	System dynamics integration step; spacecraft command; use of physical resource; parameter update; travel to new location.
Decision Action	Assess the environment and adapt to context by setting work strategy configuration variables.	Triage pending tasks; change task delegation approach; choose highest priority action.
Function	A logical collection of temporal and/or decision actions based on means-end relationships in an AH.	Land an aircraft; downlink data from satellite; plan mission timeline.
Basic Agent	A cognitive agent capable of performing work actions perfectly.	System dynamics propagator; high speed automation; spacecraft.
Performance Agent	A cognitive agent constrained by human factors such as workload, placing a cap on number of simultaneous actions.	Astronauts; operators; pilots; teams.
Function Allocation	Work configuration detailing which actors perform which actions/functions, who supervises whom, and how long actions take to perform. Ingested at runtime.	FA 1: Activity planning is allocated to human team. FA 2: Activity planning is allocated to onboard automation.

2. Methodology

Typically, WMC is used by first conducting a WDA of a work domain to generate an AH. The AH is then used to generate a work model, as it details constraints, actions, and resources in the work domain. Next, scenarios are developed that specify participating agents, function allocations, and initial conditions. Finally, simulations are run, and scenarios and function allocations are adjusted as needed as the modeler explores trade spaces and discovers new ConOps.

This work took a different approach by treating WMC as a validation tool. The WDA and AH were bypassed, as the RTCM work domain was already well documented in operating procedures, activity development materials, reports, contact recordings, telemetry, and software documentation. First the work model was created using these resources and incrementally tested using a basic scenario with one agent. After the work model was finalized, a scenario was generated to match the flight RTCM implementation. Finally, increasingly realistic FAs were developed to replicate the RTCMs as performed in flight.

2.1 Work Model

2.1.1 Work Environment

In WMC, work is contextual and is shaped by the environment in which it is performed. The work environment is represented by a collection of tangible resources that can be read and manipulated by actions. Any work environment could be expanded to a seemingly infinite number of resources of increasingly fine resolution, but such detail comes with complexity and added development time. There is little reason to include a resource that is not expected to change over the course of the simulation. In this work, resources were implemented as needed during action development, ensuring that only essential resources were modeled, as shown in Table 2.

Table 2. LF work model resources.

Information Resource	Description	Data Type
Quaternion	Spacecraft quaternion, used for attitude kinematics and spacecraft state vector.	Double x4
Commanded quaternion	Input quaternion for attitude controller, determined by ADCS.	Double x4
Quaternion error	Difference between quaternion and commanded quaternion.	Double x4
Body slew rates	Spacecraft rotation rates in spacecraft body frame, part of spacecraft state vector.	Double x3
Reaction wheel momentum	Angular momentum of each reaction wheel; used for dynamics and part of spacecraft state vector.	Double x3
Wheel control torque	Set by attitude controller, primary method of changing attitude.	Double x3
Total momentum	Total spacecraft momentum, including body rates and wheel momentum, in spacecraft body frame.	Double x3
Burn momentum	Total spacecraft momentum converted to burn frame, in which thrust torque opposes rotation torque at key phase $\theta = 180°$.	Double x3
Momentum magnitude	Magnitude of burn momentum vector in momentum circle plane, used for calculating setup and takedown burns.	Double
Target momentum	Target momentum magnitude based on estimated thrust force.	Double
Momentum bounds	Defines bounds of allowable momentum magnitude. Maneuver must be adjusted if bounds are violated.	Double x2
Phase	Angle of projected momentum vector about origin of burn plane.	Double
Active thruster	Which thruster is active, 1-4. Used to calculate control attitudes and burn momentum frame transformations.	Integer
Estimated thrust	The estimated force of the thruster based on Doppler residuals.	Double
Actual thrust	The actual force of the thruster. Can change mid simulation.	Double
Thruster on	Whether or not the thruster is exerting a torque.	Boolean
Estimated latency	Latency as measured by a latency calibration.	Double
Actual latency	Actual latency between ground and LF.	Double
Progress	Progress through the RTCM flowchart: start, setup, main, mainstop, takedown, and complete.	String
Pointing mode	ADCS control mode: idle, setup, or rotating.	String
Rotation stop/start times	Defines times of ADCS mode transitions.	Double x2
Burn duration	Duration of primary burn.	Double
Time remaining	Defines how much time remains in the burn.	Double

2.1.2 Work Model Actions

To begin, actions were implemented directly from the RTCM flowchart shown in Fig. 6. Many steps in the flowchart did not logically constitute atomic actions, and some were divided into multiple actions in the work model. Some remained purposefully undivided, as dividing them would add unnecessary complexity to the simulation. They remain abstracted, similar to how WMC functions are treated. A more thorough work model would divide these actions into atomic ones. The resulting action list is shown in Table 3. Each action is linked to resources by get or set relationships as shown in Fig. 12. Currently, no functions are implemented in the LF work model.

Table 3. Actions implemented in LF work model. Actions marked by a D superscript are decision actions while all others are temporal.

Action Name	Description
ADCSRanging	Configure ADCS controller to inertial hold mode. First and last attitude of interest.
ADCSSetup	Configure ADCS controller to setup mode. Prepares for and ends rotation mode.
ADCSRotate	Configure ADCS controller to rotate mode.
BuildMainBurn	Calculate and queue a main burn.
BuildSetupBurn	Calculate and queue a setup burn.
BuildTakedownBurn	Calculate and queue a takedown burn.
Burn	Perform a burn.
DynamicsStep	Integration step of system dynamics.
GoodMomentum^{D}	Check to see if momentum violates bounds.
GoodThrust^{D}	Check to see if spacecraft is thrusting.
GroundProcess	Transform raw spacecraft telemetry into burn momentum, phase, and time remaining.
LatencyCalibration	Perform latency calibration.
Overshoot^{D}	Determine if spacecraft momentum is too high.
ReversePumpDecision^{D}	Decide if pump should be reversed after thruster performance drops.
ReversePump	Reverse the fuel pump and pulse thrusters.
StopBurn	Stop the current burn.
ThrusterOn	Turn the thruster on.
ThrusterOff	Turn the thruster off.
TimeRemaining^{D}	Decide if enough time remains to continue a burn.
UpdateController	Update controller setpoints and determine control torque.
UpdateThrust	Update thrust prediction and calculate new target momentum state.

2.1.3 Satellite Dynamics and Control Model

Understanding and modeling the system dynamics of LF was key to developing the RTCM process. Accordingly, a key element to an LF work model is introducing a dynamics and control model that is consistent with the real-world dynamics seen with LF. This model was created and validated in Simulink prior to implementation in WMC.

Euler's moment equations were used to implement the dynamics of a rigid-body satellite with three reaction wheels aligned along the spacecraft body frame vectors [11]. System dynamics propagated accordingly to torque inputs from either reaction wheels or thrusters. The dynamic equations were integrated numerically using a fourth-order RK4 method to improve accuracy and mitigate numerical instability during long-duration burns or periods of high angular acceleration.

Attitude kinematics were implemented using the direct quaternion multiplication method, which offers improved numerical fidelity over RK4 series integration by avoiding problematic quaternion addition and enforcing unit norm throughout integration steps [12]. This method updates the attitude quaternion using a direct computation of the exponential map of the angular velocity quaternion over a time step, ensuring consistent frame alignment even in the presence of rapid rotations or large propagation time steps [13].

Attitude control was implemented using a quaternion feedback controller based on the controller described by Bacon [14]. The control law used the error quaternion between the desired and current attitude, which was fed into a PID controller then mapped to a corrective reaction wheel torque. The control torque was not bounded, leading to unrealistically fast response times when the ADCS was commanded to new attitudes. Adding a realistic saturation limit to reaction wheel torques and speeds would better model the real-world dynamics.

In the simulation, system dynamics were propagated with a DynamicsStep action that executed every 0.01 seconds. The control parameters were updated with an UpdateController action that executed every 0.01 seconds but started 0.005 seconds into the simulation. With this implementation, the system dynamics would be stepped, then the control parameters updated, then the process repeats until the simulation is over.

2.2 RTCM Scenario

2.2.1 Actors

Actors to be included in the RTCM scenario were identified by reviewing tactical procedures followed by operators while performing RTCMs. Any human, team, automated tool, or automated spacecraft subsystem referenced in the procedure was eligible to be included in the simulation. To reduce complexity, the candidates were reduced to the five essential agents shown in Table 4.

Table 4. Actors in RTCM scenario.

Agent	Description	Agent Type
MOC	A team of two operators: flight director and flight controller. Responsible for executing procedures, coordinating communication, using SMARTs, and making work strategy decisions. While the two operators could be modeled independently, complete with authority structures and associated teamwork functions, for this analysis they were abstracted to a single agent to reduce complexity.	Performance
MDNAV	A member of JPL's MDNAV team. After each thruster firing, MDNAV looks at the Doppler residuals from DSN telemetry, and provides the total ΔV to the MOC to calculate thrust force.	Performance
Prop	The LF propulsion team. If a thruster were to drop thrust mid burn, MOC would poll Prop for a decision on the next action to take.	Performance
LF	The Lunar Flashlight spacecraft could have been modeled as two separate pieces of automation relevant to RTCMs, the ADCS and propulsion system. However, such a distinction adds unnecessary complexity to the model without further insight.	Basic
SMARTS	The SMARTS application used by MOC to process telemetry, build commands, and queue them to be radiated at precise times for execution at a target phase.	Basic

MOC, MDNAV, and Prop were designated as Performance agents, which unlike basic agents, have unique internal dynamics that constrain number of concurrent tasks to model agent workload and action delays when an agent is too busy to perform an action. Currently none of these capabilities are utilized during simulation, as the ground-in-the-loop RTCM procedure were designed to minimize the number of concurrent tasks that operators would perform.

2.2.2 Function Allocations

This work presents simulation results from two different FAs using the same work model and scenario. Since the WMC simulation is pre-compiled and passed FAs as .csv files at runtime, identifying the differences between these two FAs requires no additional programming. The two function allocations are detailed in Table 5.

FA 1 models ground-in-the-loop RTCMs as they were performed in flight. Each action has an executing agent that performs the action, and a responsible agent that schedules the action to be taken. For example, the MOC presses a button on SMARTS that commands it to perform a BuildMainBurn action. After the MOC approves the action, SMARTS sends the Burn Command to LF. Durations for FA 1 were chosen to be close to those seen in videos of the two ground-in-the-loop RTCMs.

FA 2 models a hypothetical case in which LF had an accelerometer and command sequences that supported branching logic, telemetry conditionals, loops, and other functions required to place all closed-loop control of RTCMs onboard LF. Note that in this FA, the actions have no responsible agent. There is no functional difference between an action having no responsible agent and an action having the same executing and responsible agent.

Table 5. Function allocations for each RTCM scenario.

Action	FA 1			FA 2		
	Executing Agent	Responsible Agent	Duration (s)	Executing Agent	Responsible Agent	Duration (s)
ADCSRanging	LF	None	10	LF	None	10
ADCSRotate	LF	None	300	LF	None	300
ADCSSetup	LF	None	10	LF	None	10
BuildMainBurn	SMARTS	MOC	5	LF	None	0
BuildSetupBurn	SMARTS	MOC	5	LF	None	0
BuildTakedownBurn	SMARTS	MOC	5	LF	None	0
Burn	LF	SMARTS	Variable	LF	None	Variable
DynamicsStep	LF	None	0	LF	None	0
GoodMomentum	SMARTS	MOC	1	LF	None	1
GoodThrust	SMARTS	MOC	1	LF	None	1
GroundProcess	SMARTS	LF	0	LF	None	0
LatencyCalibration	LF	MOC	10	LF	None	0

Overshoot	SMARTS	MOC	10	LF	None	1
ReversePump	LF	MOC	10	LF	None	4
ReversePumpDecision	Prop	MOC	30	LF	None	1
StopBurn	LF	MOC	15	LF	None	1
ThrusterOff	LF	None	1	LF	None	1
ThrusterOn	LF	None	1	LF	None	1
TimeRemaining	SMARTS	MOC	1	LF	None	1
UpdateController	LF	None	0	LF	None	0
UpdateThrust	MDNAV	MOC	30	LF	None	1

2.2.3 Communications Delays

One of the primary challenges to ground-in-the-loop RTCM execution is the light-time delay on command execution, which in flight varied between 4 and 30 seconds depending on LF's location. This challenge was the primary obstacle to ground-controlled RTCMs and was the sole reason for the latency calibration action performed prior to every RTCM [15]. It is essential that an RTCM WMC simulation models the same light-time delay.

In the RTCM scenario, communication time delays were defined between each agent. The communications delay from LF to Earth-based agents was 12 seconds, the delay between human teams was given 3 seconds to represent challenges of telework, and the delay between SMARTS and MOC is 0 seconds, as the MOC agent is actively using the tool.

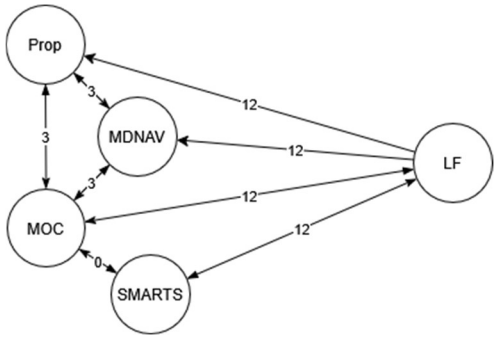

Fig. 7. Communications delays between simulation actors.

For each action, function allocations must specify an executing agent and may specify a responsible agent. For the RTCM scenario, the responsible agent schedules an action for the executing agent to perform. The WMC simulation engine will automatically apply communication delays to actions with differing executing and responsible agents. For example, a GoodThrust action executed by MOC decides that thrust has died and immediately schedules a StopBurn action for LF. The simulation engine will add 12 seconds to the execution time of StopBurn to account for communications delay. The LatencyCalibration command estimates this communications delay, allowing actions like BuildMainBurn to account for it when timing actions to execute at a target phase.

2.2.4 Thruster Behavior Timeline

The scenario considered in this paper models a realistic RTCM scenario in which a thruster dies in the middle of a 10-minute burn but is refreshed after a reverse pump maneuver. To show this, the events shown in Table 6 are injected into the scenario. The effects of the events on RTCM performance are revealed by the system behavior under each function allocation.

Table 6. Thruster behavior in the RTCM scenario.

Event	Description
Start of scenario	From previous contacts, operators expect a thrust force of 200 mN. However, the thruster is performing at 250 mN.
$t = 5\ min$	FOD blocks thruster line causing thrust to drop to 50 mN. Burn is aborted with no pump reversal.
Burn abort	After burn is aborted, FOD worsens and thrust drops to 0 mN. This prompts a pump reversal at the next pump reversal decision point.
Pump reversal	After the pump is reversed, the thruster is recovered and thrust improves to 120 mN for the rest of the burn.

3. Results

While a WMC simulation is running, a runtime log displays actions as they are performed in the simulation, as shown in Fig. 8. Additionally, interactions between agents, action traces, resource changes, and other relevant information are logged to the filesystem, and can be used to generate plots and figures after the simulation has concluded.

```
t=0.000000: LF performs FlashlightWMLatencyCalibration, duration = 10.000000
                Setting latency to 24.000000s
t=0.000000: LF performs FlashlightWMADCSRanging, duration = 10.000000
t=10.000000: LF performs FlashlightWMADCSSetup, duration = 10.000000
t=20.000000: LF performs FlashlightWMADCSRotate, duration = 600.000000
t=25.000000: SMARTS performs FlashlightWMBuildSetupBurn, duration = 5.000000
                Current momentum: 0.000983
                Target momentum: 0.225624
                Momentum is low, setup burn required
                Momentum is low, send ASAP
                Queueing a 9.958403s burn after 0.000000s
                MOC schedules FlashlightWMBurn for LF after 5.000000 + 12.000000s
                Scheduling a momentum check after 38.958403s
                MOC schedules FlashlightWMGoodMomentum for SMARTS after 38.958403 + 0.000000s
t=42.000000: LF performs FlashlightWMBurn, duration = 0.000000
                Current phase: 252.417083
                Scheduling thruster on
                Scheduling thruster off
t=42.000000: LF performs FlashlightWMThrusterOn, duration = 1.000000
t=51.958403: LF performs FlashlightWMThrusterOff, duration = 1.000000
t=63.9584: GoodMomentum chooses option Momentum outside threshold - setup phase.
                Momentum magnitude: 0.271497
                Momentum bound low: 0.215624
                Momentum bound high: 0.235624
                Scheduling overshoot check
                MOC schedules FlashlightWMOvershoot for SMARTS after 1.000000 + 0.000000s
t=64.9584: Overshoot chooses option Momentum has overshot.
t=64.958403: Overshoot segment momentum bad
                Scheduling UpdateThrust
                MOC schedules FlashlightWMUpdateThrust for MDNAV after 10.000000 + 3.000000s
t=77.958403: MDNAV performs FlashlightWMUpdateThrust, duration = 30.000000
                Setting in setup or start segment
                Setting thrust to actual thrust: 0.250000
                MOC schedules FlashlightWMBuildSetupBurn for SMARTS after 30.000000 + 0.000000s
```

Fig. 8: WMC simulation runtime log. Custom logging messages for all actions have been defined.

3.1 Function Allocation 1

FA 1 attempts to mimic how ground-in-the-loop RTCMs were performed in flight and is shown in Fig. 9. With the challenges introduced by a 24-second round-trip light-time delay, the actions are more delayed than those in FA2. FA 1 resulted in two minutes of burning at nominal thrust and one minute at off-nominal thrust. While the first setup burn was executed perfectly, the MOC did not have ample time to respond to the variable thruster performance, and ended the RTCM with a takedown burn that only brought the spacecraft's momentum state halfway to zero, rather than all the way.

Note that in Fig. 9 the MOC does not appear as an agent. Although the MOC is the responsible agent for most of the actions in FA 1, the MOC agent is not an executing agent for any action. Instead, the MOC delegates actions to other agents in the system and is therefore not represented in the figure.

3.2 Function Allocation 2

FA 2 represents the hypothetical case where all capabilities and authorities were assigned to LF. With only one actor in the simulation, there was no light-time delay on communications, allowing for more agile responses to changes in the system. As a result, the RTCM contains over six minutes of nominal thrusting; over three times that of FA 1. The recovery from changes in thruster performance was so rapid that LF was able to perform a second main burn over two minutes long and still finish in time for a complete takedown burn, returning the spacecraft to its original momentum state. To little surprise, RTCMs would have been far more efficient had LF been able to execute them autonomously.

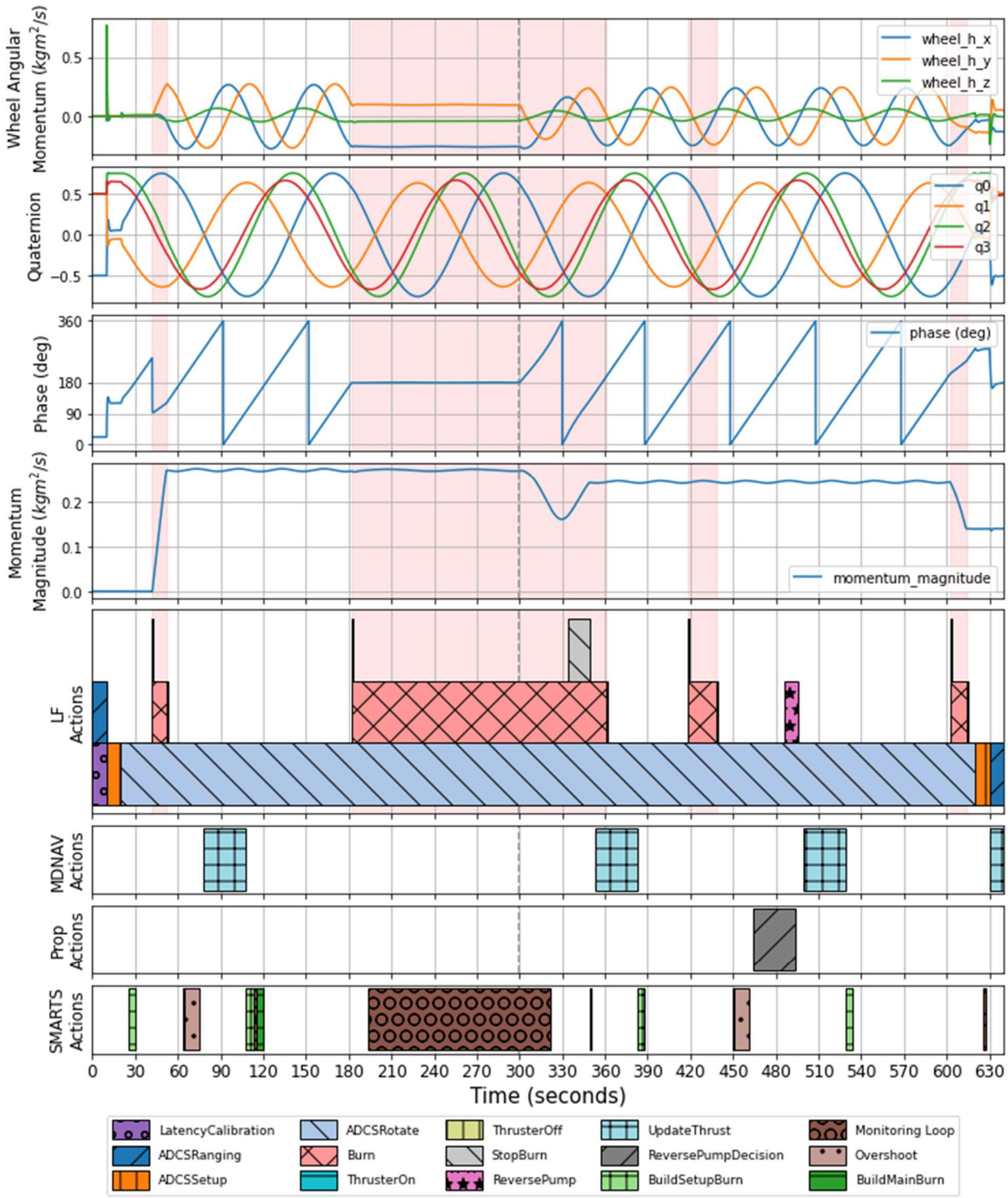

Fig. 9. System timeline for Function Allocation 1. The red shaded regions indicate times where the thruster is firing. The vertical dashed line indicated the moment when thrust is lowered to 50 mN. Action timelines are divided by executing agent.

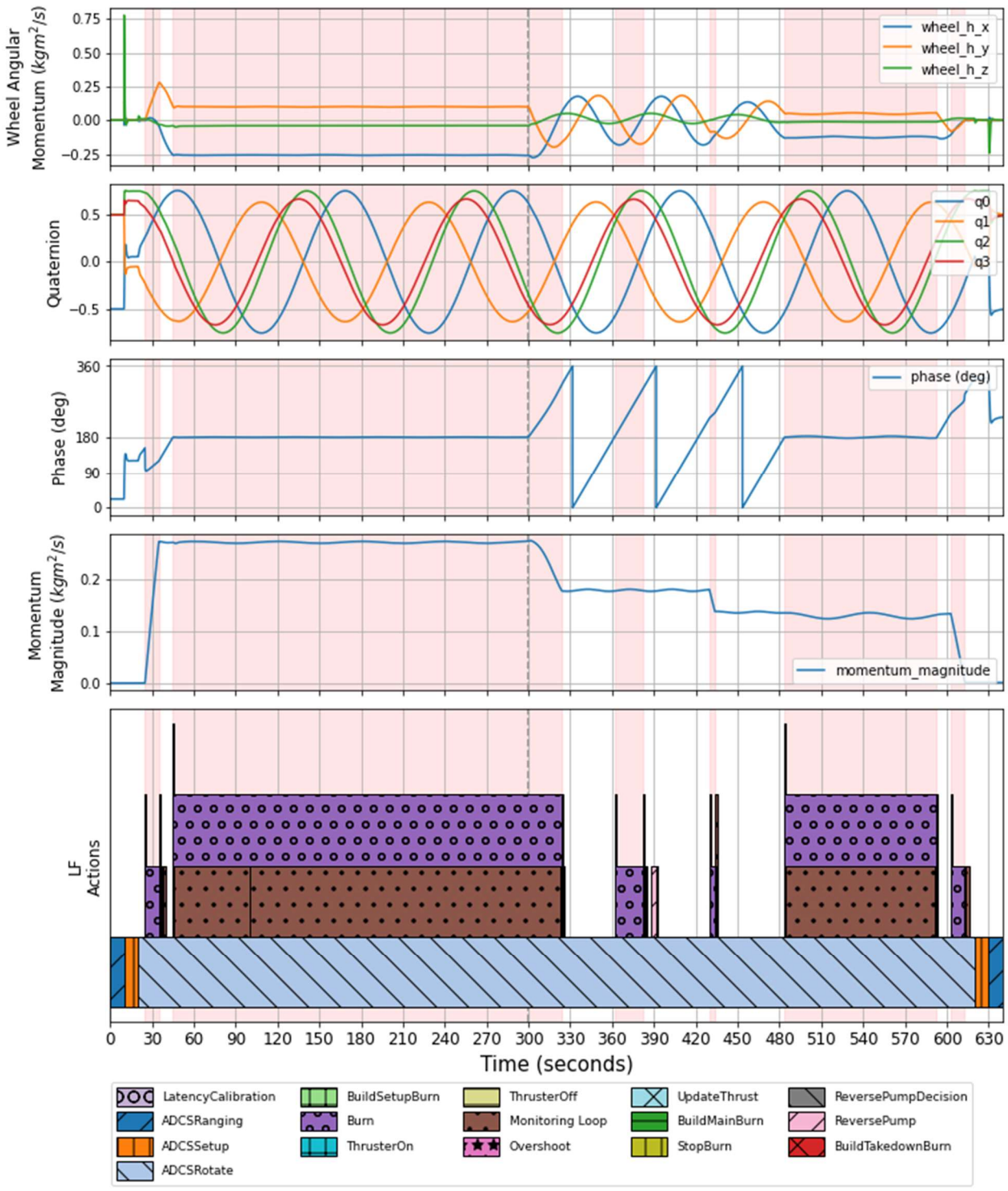

Fig. 10. System timeline for Function Allocation 2. The red shaded regions indicate times where the thruster is firing. The vertical dashed line indicated the moment when thrust is lowered to 50 mN. Action timelines are divided by executing agent.

4. Discussion

4.1 Comparison to Flight Data

Ground-in-the-loop RTCMs were attempted only twice before thrust ceased completely; in both instances, the thrust was extremely low—almost negligible. A more relevant comparison is provided by a 20-minute burn attempted during Contact 87, in which the thrust dropped significantly halfway through the burn (see Fig. 11). This RTCM was commanded in open-loop via a command sequence while operators monitored in real time. The quaternion telemetry is comparable to that of the WMC simulation, though slight differences exist because this burn was executed on thruster four, whereas the simulation was performed on thruster three.

LF began with a near-zero momentum state, as indicated by the reaction wheel telemetry, then slewed to the setup attitude and initiated rotation. The thrust observed during the setup burn deviated from the estimated thrust used to generate the command sequence, resulting in a momentum state after the setup burn that differed slightly from the intended state. Consequently, low-amplitude oscillations in the wheel rates were observed during the first half of the main burn. These oscillations are absent in the WMC simulation because the UpdateThrust action perfectly estimates thrust without error, resulting in wheel RPM oscillations of near-zero amplitude.

When the thrust fell off halfway through the burn, the amplitudes of the wheel RPMs increased dramatically. The onset of this behavior is visible in Fig. 9 after the thruster deactivated; however, since the burn was stopped shortly thereafter, the long-term effects are less pronounced. During Contact 87, operators also disabled the propulsion system via real-time command shortly after the thrust change to prevent the spacecraft momentum state from drifting into unsafe ranges. Without SMARTS, operators had no means to command further real-time propulsion activities, so they observed only the final 10 minutes of sequenced rotation.

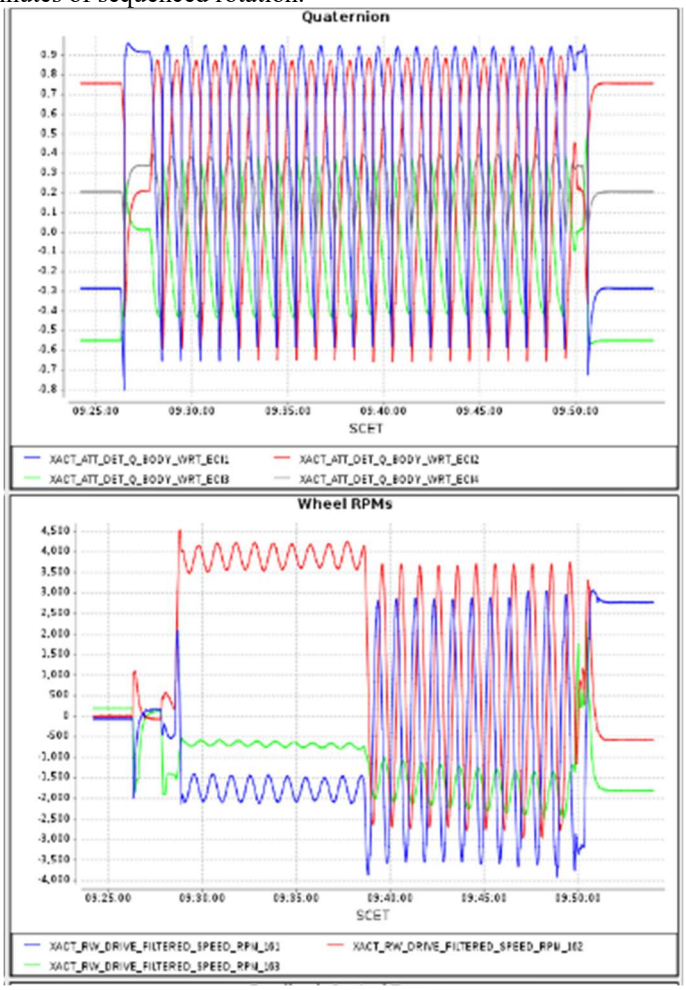

Fig. 11. Spacecraft telemetry from open-loop RTCM attempted during contact 87.

4.2 Future Work

A primary goal for this project was to leverage WMC's ability to automatically generate and model teamwork actions when actions are delegated from one agent to another. Commanding, confirmation, monitoring, and taking direct control are all teamwork actions that were present in RTCM execution but are currently omitted from the WMC simulation. For example, when the MOC commanded SMARTS to perform a BuildSetupBurn action, there was a confirmation action afterwards in which the MOC checked the generated setup burn parameters before radiating the commands. After the commands were sent, MOC would monitor spacecraft telemetry while the burn executed, and confirm its successful completion. All of these teamwork actions take time and thus impact the RTCM timeline. Had these teamwork actions been implemented, the simulation would more closely resemble RTCMs as they were performed in flight. Additionally, the generated timelines show that the MOC was in fact performing actions during RTCMs; they did not simply delegate the whole time with no additional effort.

This project considered only two function allocations: the flight implementation case and that of full spacecraft autonomy. The results are unsurprising; full spacecraft autonomy performs better than how RTCMs were performed in practice. The original development ground-in-the-loop RTCMs was driven by system constraints, most importantly the limitations of LF's autonomy. Future investigations could identify an optimal function allocation that still complies with the constraints faced in flight, to show how the RTCM process could have been improved operationally.

4.3 Lessons Learned

Aside from the high-level observation that having the ability to dynamically adapt spacecraft autonomy can improve mission performance in the face of unpredictable events, this project yielded valuable lessons learned for WMC developers.

WMC is not a validation tool. It is designed to discover novel concepts of operations for unique systems of multiple cognitive agents; not replicate the intricate details of a bespoke propulsion technique for a well-known system in which most of the interactions take place between just two agents. Most of the time spent working on this project was spent on polishing the minute details that make RTCMs possible, such as replicating the logic of the ADCS controller or implementing latency corrections. While WMC can replicate those details to model a system at a high level of fidelity, its strength lies in revealing the emergent ConOps of work configurations to optimize FAs. This project modeled a known system and worked towards a known end, rather than modeling new systems to reveal novel ends. Future WMC modelers should not attempt to use WMC as a validation tool.

This project bypassed the initial WDA and development of an AH, assuming that the procedures put in place by LF operators would be suitable work model actions. They were not; the actions laid out in LF procedures were not atomic and simplified information exchanges to accelerate tactical operations. As a result, many of the actions in the RTCM work model are ambiguous and oversimplified. For example, the UpdateThrust action is modeled as a simple update to an estimated thrust parameter, commanded by the MOC and performed by the MDNAV agent. In practice, this action included MDNAV measuring doppler residuals in the real-time DSN telemetry and running a script that used residuals and trajectory ephemeris to calculate thrust estimates. The MOC would then poll MDNAV for the thrust estimate, confirm the value, and enter it into SMARTS. This process represents seven distinct interactions between agents, rather than the single interaction considered in this work model. Had a WDA been performed at the start of this project, actions would have been implemented as logical building blocks that could truly leverage the strengths of WMC.

5. Conclusions

This study successfully applies WMC to the complex domain of satellite operations, using LF RTCMs as a case study. By leveraging the foundational principles of Work Domain Analysis (WDA), WMC captures context and decouples work from the agents, allowing users to simulate a range of function allocations and capture the emergent behaviors that characterize real-world operations. The strength of WMC lies not in replicating minute details of bespoke systems, but in its ability to reveal emergent operational concepts and inform trade-space decisions in the design of multi-agent socio-technical systems.

The LF mission used RTCMs to perform propulsive maneuvers with only a single thruster. Due to thruster unpredictability, operators needed to close the control loop by commanding the spacecraft in real time during propulsive maneuvers. Bespoke software and operational flows were developed to enable this mode of operation under constraints of light-time delays, human response times, and limited spacecraft autonomy. It was determined through testing different function allocations in WMC that allocating work functions to spacecraft autonomy results in superior RTCM performance.

This project validates WMC as a valuable tool for early ConOps design and mission operations analysis, providing early insights into novel work domains, human-automation interactions, and function allocation strategies.

Acknowledgements

I sincerely thank my advisors, Dr. Glenn Lightsey and Dr. Karen Feigh, for their continued guidance and support as I forge my path through academia.

This work would not have been possible if not for the relentless efforts of the LF team in pursuit of recovering their beloved spacecraft, including the GT operations team, JPL MDAV, JPL engineering support, JPL management, the LF science team, and LF propulsion team.

Special thanks go to Abhi Paladugu, for helping a stranger across the country learn the ropes of WMC.

Appendix A

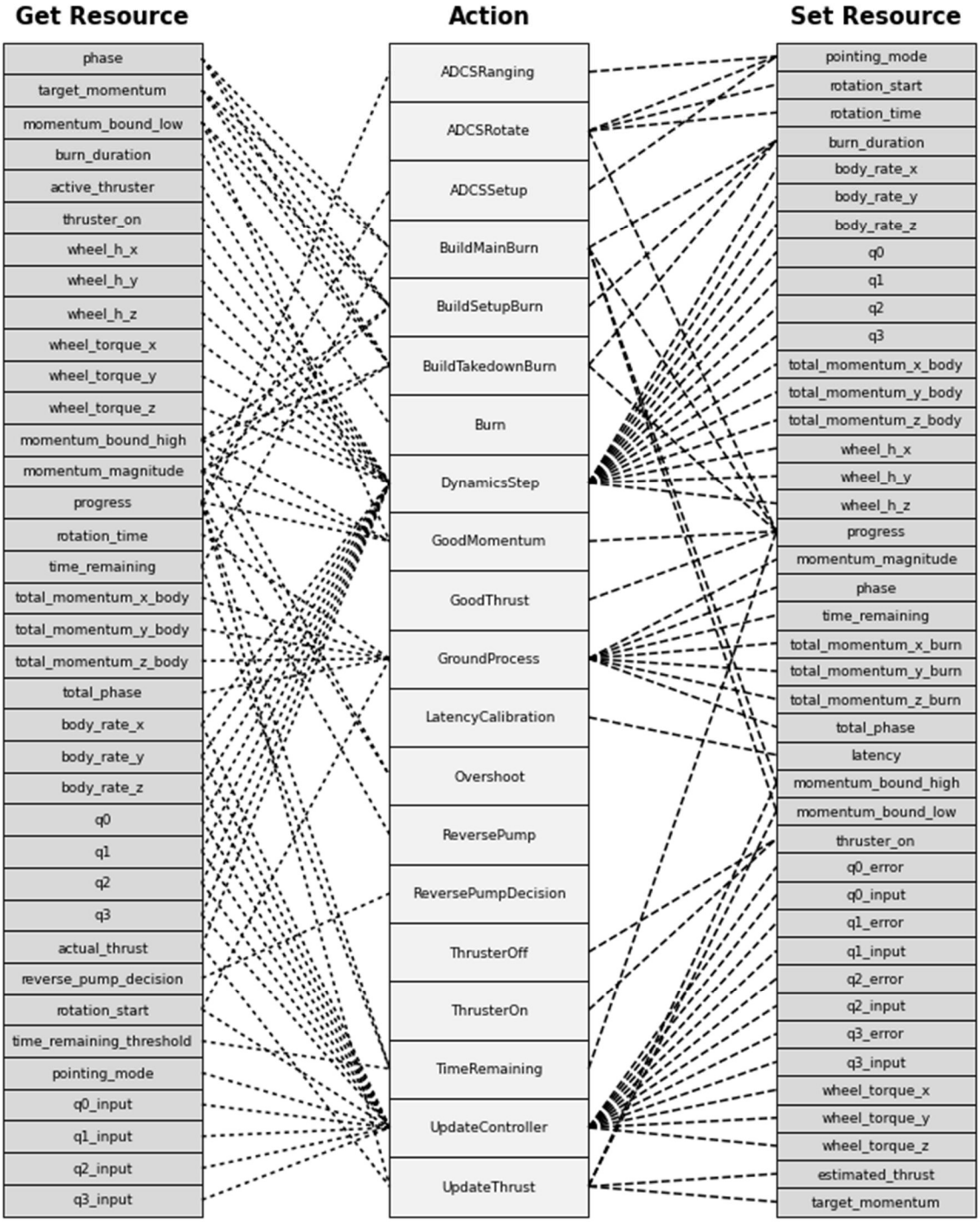

Fig. 12. Work model actions linked to the resources that they get and set.

References

[1] N. Naikar, R. Hopcroft, and A. Moylan, "Work Domain Analysis: Theoretical Concepts and Methodology," Defence Science and Technology Organisation Victoria (Australia) Air Operations Div, ADA449707, 2005. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA449707

[2] M. Starr, M. Hauge, and E. G. Lightsey, "Shining a Light on Student-Led Mission Operations: Lessons Learned from the Lunar Flashlight Project," in *Small Satellite Missions and Mission Concepts*, Jan. 2024. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2024-0822

[3] B. A. Cohen, P. O. Hayne, B. Greenhagen, D. A. Paige, C. Seybold, and J. Baker, "Lunar Flashlight: Illuminating the Lunar South Pole," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 35, no. 3, pp. 46–52, Mar. 2020, doi: 10.1109/MAES.2019.2950746.

[4] C. Smith *et al.*, "The journey of the lunar flashlight propulsion system from launch through end of mission," in *Proc. Small satellite conference*, in SSC23-VI-03. 2023.

[5] T. McElrath, S. Collins, K. Lo, C. Smith, N. Cheek, and M. Hauge, "A delicate balance of torque and thrust: How lunar flashlight used rotating maneuvers to make one thruster do the work of four," in *AAS/AIAA astrodynamics specialist conference*, 2023. [Online]. Available: https://www-robotics.jpl.nasa.gov/media/documents/ASC-23-260.pdf

[6] M. Hauge *et al.*, "Operations Systems Engineering for the Lunar Flashlight Mission," in *Year in Review - Research & Academia*, Aug. 2023.

[7] A. R. Pritchett, K. M. Feigh, S. Y. Kim, and S. K. Kannan, "Work Models that Compute to Describe Multiagent Concepts of Operation: Part 1," *J. Aerosp. Inf. Syst.*, vol. 11, no. 10, pp. 610–622, Oct. 2014, doi: 10.2514/1.I010146.

[8] K. M. Feigh, A. R. Pritchett, S. Mamessier, and G. Gelman, "Generic Agent Models for Simulations of Concepts of Operation: Part 2," *J. Aerosp. Inf. Syst.*, vol. 11, no. 10, pp. 623–631, Oct. 2014, doi: 10.2514/1.I010147.

[9] M. IJtsma, L. M. Ma, K. M. Feigh, and A. R. Pritchett, "Demonstration of the 'Work Models that Compute' Simulation Framework for Objective Function Allocation," *Proc. Hum. Factors Ergon. Soc. Annu. Meet.*, vol. 62, no. 1, pp. 321–324, Sep. 2018, doi: 10.1177/1541931218621074.

[10] A. R. Pritchett, S. Y. Kim, and K. M. Feigh, "Measuring Human-Automation Function Allocation," *J. Cogn. Eng. Decis. Mak.*, vol. 8, no. 1, pp. 52–77, Mar. 2014, doi: 10.1177/1555343413490166.

[11] M. Blanke and M. B. Larsen, "Satellite Dynamics and Control in a Quaternion Formulation - Lecture note for course 31365 Spacecraft Dynamics and Control at DTU", [Online]. Available: https://backend.orbit.dtu.dk/ws/portalfiles/portal/98594729/Satdyn_mb_2010f.pdf

[12] S. A. Whitmore, "Closed-form integrator for the quaternion (euler angle) kinematics equations," May 09, 2000 Accessed: Apr. 05, 2025. [Online]. Available: https://ntrs.nasa.gov/citations/20080004113

[13] M. Hauge, "Attitude Determination from Angular Velocity and Simple Sensor Inputs for Gravity-Gradient-Stabilized CubeSats".

[14] B. Bacon, "Quaternion-Based Control Architecture for Determining Controllability/Maneuverability Limits," in *AIAA Guidance, Navigation, and Control Conference*, Minneapolis, Minnesota: American Institute of Aeronautics and Astronautics, Aug. 2012. doi: 10.2514/6.2012-5028.

[15] M. Starr, "Development of Tactical and Strategic Operations Software for NASA's Lunar Flashlight Mission," in *Frank J. Redd Student Competition*, Aug. 2023. [Online]. Available: https://digitalcommons.usu.edu/smallsat/2023/all2023/4/