Event-Driven Modeling and Execution of Robotic Activities and Contingencies in the Europa Lander Mission Concept Using BPMN

Jean-Pierre de la Croix[1], Grace Lim[1]

[1]Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109, USA,
Jean-Pierre.de.la.Croix@jpl.nasa.gov, Grace.Lim@jpl.nasa.gov

ABSTRACT

TRACE (Traceable Robotic Activity Composer and Executive) is a tool designed to address the modeling, verification, and execution of planned and contingent activities during robotic space missions. These activities can be modeled (i.e., planned) graphically in TRACE using the Business Process Model and Notation (BPMN) language. This standardized language is used to model a sequence of robotic activities tied together by logical constructs, events, and time. TRACE includes verification tools to ensure feasibility before mission models are executed by TRACE's executive, which is integrated into the autonomy subsystem via a connector. In this paper, we describe TRACE in detail as it would apply to a conceptual Europa Lander surface mission with planned excavation and sample collection activities, as well as, contingency activities when execution is off-nominal—all highlighting our novel use of BPMN as a lingua franca for the pipeline from mission planning to autonomous execution.

1 INTRODUCTION

Often planning of robotic activities for space (including planetary, small body, etc.) exploration focuses on generating activity timelines and relying on replanning to opportunistically target secondary mission objectives or respond to non-critical events [1]. On-board activities are then executed according to the timeline. An alternative approach is to model and execute the mission as a sequence (or flow) of activities with decisions to "replan" (i.e., flow to another activity) being driven by events. TRACE (Traceable Robotic Activity Composer and Executive) is a tool designed to holistically address the modeling, verification, and execution of planned, opportunistic, and contingency activities during robotic missions from an event-driven execution perspective.

The *Business Process Modeling and Notation* (BPMN) language is a standard maintained by the Object Management Group that graphically models business processes [2]. Elements within BPMN include events based on state or data, tasks for humans or automation, logical gateways for process flows, and pools and lanes for organizing resources (see reference guide for more details [3]). TRACE (formerly M2PEM [4]) tailors BPMN to the robotics domain. Automated tasks, like service tasks, correspond to robotic activities, like navigation or grasping an instrument with a robotic arm. Data-driven elements, like conditional events or exclusive gateways, use system data to allow the executive to make decisions on how to flow through the mission. TRACE follows the BPMN rule set and integrates these elements with the robotic system. Consequently, this standardized language can be used to model a sequence of robotic activities tied together by logical constructs, events, and time. A BPMN mission model encodes the sequence of activities for the mission. The BPMN event constructs provide a way to encode responses to opportunities (e.g., detection of a new, interesting science target) or non-deterministic events (e.g., a subsystem fault) and diverge the process flow to a secondary process that contains a sequence of alternate activities. These contingent (or opportunistic) activities can be modeled to execute in parallel to planned activities (e.g., start up an instrument) or to interrupt and then resume planned activities once completed. Since TRACE's modeling tool (composer) allows a large degree of flexibility in designing the mission, TRACE provides verification tools to ensure feasibility. Specifically, the BPMN mission models can be translated into the Process Meta Language (PROMELA) and input into the Spin Model Checker with a set of mission requirements expressed in Linear Temporal Logic (LTL). Consequently, TRACE is able to verify if the mission completes (e.g., no dead locks, no starvation) and if it satisfies the mission requirements (e.g., perform science measurements at least k times) [4]. Lastly, verified mission models are then executed by TRACE's executive, which integrates into the autonomy subsystem via a connector. The executive accesses data (e.g., sensors, system health) and invokes subroutines (e.g., navigate to coordinate) according to the mission model by interpreting it at runtime.

Consequently, TRACE is a potential end-to-end solution for flight missions by providing integrated tools for modeling, verification, and execution of planned, opportunistic, and contingency robotic activities. In this paper, we describe TRACE in detail as it would apply to a conceptual Europa Lander surface mission, with planned excavation and science activities, as well

as, contingency activities in response to subsystem faults--all highlighting our novel use of BPMN as a lingua franca for the pipeline from mission planning to autonomous execution.

TRACE has previously been demonstrated in missions with multiple UxV (uncrewed vehicles) for the Office of Naval Research [4], but this paper details an effort to improve this tool for Jet Propulsion Laboratory (JPL) and NASA use in planetary robotic missions. It could allow NASA/JPL to include opportunistic, normally unplanned, activities with the ability to explicitly model what will happen in response to an unplanned event and to verify, prior to execution, that such a response does not compromise the overall mission. Specifically, this paper describes our first efforts to adapt TRACE to the current Europa Lander Autonomy Prototype (ELAP). We will focus our discussion on the integration of TRACE with this prototype to directly command a notional Europa Lander in simulation by creating a plugin into our existing mission executive. A plug-in (or "connector") is the primary method for integrating TRACE with another autonomy subsystem. While the executive is responsible for executing the logic of the BPMN mission model, a connector is responsible for accepting a parameterized task and invoking and monitoring the appropriate system calls. For example, if the mission model contains a task *Preheat Arm* with a target temperature in Celsius, then once the mission reaches this task, the executive will invoke the connector to set the heaters to provided target temperature. Effectively, the connector code commands the heater controllers with set temperature and monitors controller outputs to determine if the activity is complete. Once the system has completed its activity (e.g., heaters have reached the target temperature), the connector informs the executive that this task is complete. Upon completion (or failure) of a task, the mission continues as it is modeled. In this paper, a task in the mission model corresponds to an activity request that is passed to ELAP for execution. ELAP will reply with either success or failure of this sequence to the executive via the connector. In addition to system calls for activities, the connector also provides access to system data that may be fetched into mission model during runtime. Typically, this functionality is used to set up events or conditions on system data, or to push data from the mission model to the system (such as a notification). Referring back to the earlier example, an alternative way to model heaters reaching a target temperature would be to create two activities, *Turn On Heaters* and *Turn Off Heaters*, and a condition event on the heater temperature. First, *Turn On Heaters* is invoked. Upon success, the mission executive monitors the condition event, which is triggered when the heaters reach the specified target temperature. This condition event is followed by *Turn Off Heaters*. TRACE through its use of BPMN can model and execute either of these scenarios easily.

2 BENEFITS

Based on our experiences of using TRACE (and thusly, BPMN) for multi-UxV (unmanned vehicles), this mission modeling, verification, and execution tool has a number of cross-cutting benefits that would apply equally to space missions.

1. *Event-driven*: Compared to timeline sequencing, mission models can include fully modeled responses to opportunistic events or contingencies to faults that are handled by the executive directly at the time of the event.
2. *Flexibility*: Reusable atomic tasks and hierarchical task organization reduces the modeling (or remodeling) effort when mission objectives are altered.
3. *User Friendly*: BPMN is a visual, symbolic language with well-known logical constructs and easy-to-follow rules.
4. *Verifiability*: BPMN is encoded as XML and follows a rule set that can be translated to PROMELA for use with the Spin model checking tools.
5. *Standardization*: TRACE supports the BPMN 2.0 standard over relying on a custom language.

To demonstrates TRACE in its new domain, we investigated its application to the Europa Lander mission concept, but first we describe some new TRACE functionality that allows us to integrate it with other autonomy subsystems, as well as, provide visualization of the execution timeline.

3 FRAMEWORK

The internal mechanisms by which TRACE ingests BPMN mission models and executes mobility and payload behaviors have been described in detail previously [4]. At a high level, BPMN mission models are encoded in XML and passed to the executive. This executive starts with the primary process and spawns tokens for all start events. Tokens flow along

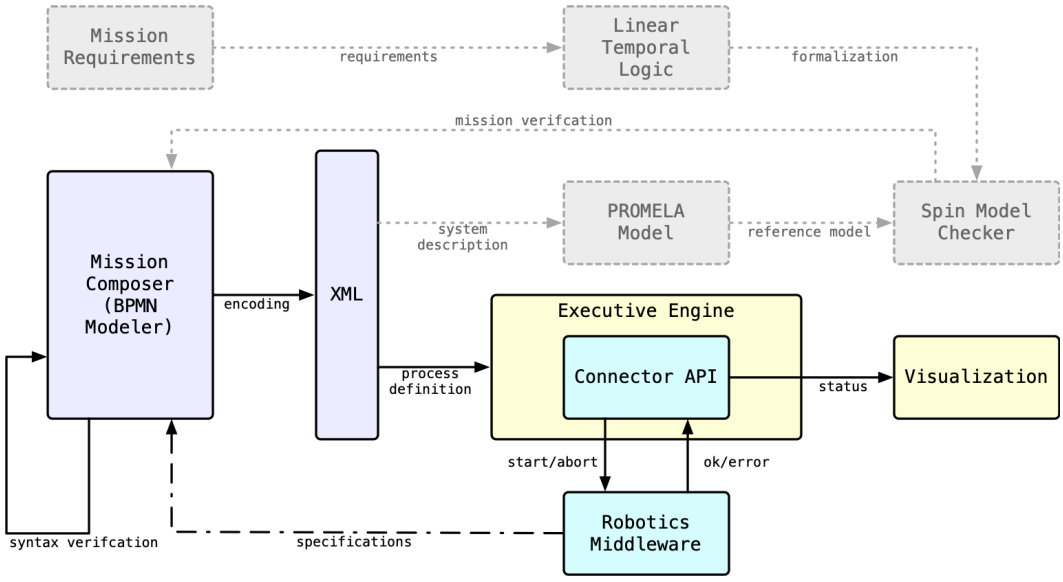

Figure 1: Improved functional architecture that uses a connector (plugin) to interface with the robotics middleware and other external components.

sequence flows and take an action at each BPMN element. For example, at parallel gateways new tokens are spawned for each outgoing sequence flow or at intermediate throw events an event of a specified type (e.g., signal or message) is emitted. At service tasks (a special type of BPMN activity), a call to the system (e.g., a robotics middleware like the Robotics Operating System (ROS) [5]) is made via a *connector*.

3.1 Connectors

A *connector* is a new feature of TRACE. Whereas the TRACE executive (formerly M2PEM [4]) used to be tightly integrated with ROS, a plug-in architecture allows TRACE to be used with more than one system. Integration with a new robotics middleware, for example, only requires the development of a new connector. Figure 1 illustrates this new architecture.

Consequently, the executive is primarily responsible for executing the BPMN mission model according to the BPMN rules, while any system-related operations are facilitated by the connector. The requirements on the connector is to provide a service-like API to activities. The executive will ask the connector to execute a task with properties (i.e., inputs) from the mission model and requires that the connector inform the executive if the task is either done or has encountered an error (or exception). The executive must also have the ability to abort the task, for example, when a boundary event requires that a service task is interrupted and terminated. The connector may also provide other system-level functionality, such as logging, an alternate clock, or message passing. A convenient side-effect of this connector-based architecture is that the executive can load multiple connectors and thus execute tasks on multiple different middlewares at once.

3.2 Europa Lander Autonomy Prototype

The Europa Lander Autonomy Prototype (ELAP) is a software testbed to explore what capabilities could be required for a potential autonomous surface mission on Europa (and other planetary surfaces). It provides a simulated platform with a world model database and controllable components typical of current robotic platforms for planetary surface missions, such as a robotic arm, instruments, batteries, and heaters. Integration of ELAP with TRACE follows the new connector paradigm.

The current iteration of the prototype uses the Robotic Operating System (ROS) to provide a service-like API through actionlib [6]. The actionlib library allows a ROS developer to provide interaction with a software component in terms of a topic name, as well as, an Action message, which defines the goal, feedback, and result. Each executable activity within the prototype has a corresponding Action message defined in the message library. Consequently, we have implemented an autocoder to automatically parse the message library for Actions and create an Action client that is used inside the connector to (a) populate goals from inputs in the service tasks defined in the mission model, (b) send (and abort) goals, and (c) capture results as the output of the corresponding service task. An XML configuration file defines the set of available activities in the prototype and then is used to

instantiate the Action clients when the mission model is loaded. As a pre-processing step, the executive will iterate over the mission model and confirm with the connector if a particular activity is available for execution. If not, then the mission fails to load. This configuration file also contains a mapping of resource names to Action topics and types, such that, for example, a service task with *resource_name* set to *functional.excavate_site* will map to the ROS Action *ExcavateSite* on the topic *functional.excavate_*site in the prototype. Lastly, each input in the service task must match up with a field in the Action request message. For example, if the *ExcavateSite* message has a field called *selected_site* of type *ExcavationSite* with a field called *site_id* of type *int*, then the input in the service must be defined as *selected_site.site_id*. Any error in the input, such as a bad field name or invalid type conversion, will be caught in the connector and reported back as an error to execute the service task.

3.3 Timeline Visualization

Another improvement to TRACE is the integration of the Canvas Timeline Viewer, which is used with other planners and schedulers at the Jet Propulsion Laboratory (JPL), such as [7]. During runtime, the executive can record the execution of the mission in terms of the activities called through the connector, as well as, the state of data used to make decisions within the mission model by recording these to a JSON formatted file. Consequently, we are able to visualize the runtime execution in the easily digested timeline format.

3.4 Resuming Missions on Restart

Lastly, a third improvement to TRACE is that mission execution state is saved to disk at runtime, such that if it is rebooted (e.g., due to a system reset), we are able to resume the mission in its last execution state. Any previously executing activity is commanded again.

The following section focuses more specifically on various mission models developed for testing on the prototype: (1) the excavation of a candidate site and handling faults, and (2) responding to events based on time and energy system states. The objective is to demonstrate that the event-driven, activity-based modeling and execution of TRACE is a promising alternative to existing timeline-based planning and execution tools. It also shows the standardized, user-friendly, and flexible representation of robotic missions using TRACE's application of BPMN.

4 EUROPA LANDER SURFACE MISSION

NASA's Europa Lander is a mission concept focused on landing a robotic spacecraft on the surface Jupiter's moon Europa [8]. The lander is carrying a variety of science instruments to search for evidence of biosignatures on Europa, assess the habitability of Europa, and characterize the surface and subsurface of Europa. Due to Jupiter's constant and intense radiation, the lander has a robotic arm to excavate approximately 10cm below the surface to collect sample material protected from the radiation [9]. It must also achieve this task mostly autonomous, since two-way communication between Earth and Europa are very lengthy and the lander may have a relatively short operational life due to radiation, cold, and limited battery life. For a simplified perspective, we can summarize the Europa Lander surface operations as:

1. Upon landing, use the cameras to capture a panoramic image.
2. Determine a set of potential excavation sites around the lander from the image.
3. While enough energy is available and we have not collected and analyzed (5) samples:
    a. Select the next best excavation site based on images, input from ground, or result of any previous analysis.
    b. Prepare the arm and excavate the site to the necessary depth.
    c. Use the camera to examine the excavation site for collection targets.
    d. For each collection target: collect, transfer, process, and analyze the sample. Then, communicate the analysis if the communication window is open.
4. If energy is still available, continue to collect images, as well as, seismic/geological data.
5. Before energy expires and communication window closes, communicate any available data.

This description of the operations for the Europa Lander basic surface mission features a number of concepts: sequential operations, decisions by a ground operator, repeated operations, as well as, a contingency for a fault. Figure 1 in the appendix is a BPMN mission model that captures the Europa Lander basic surface mission with respect to the provided high-level description in its current prototyped state. Actual flight mission models may be more complex, for example, to capture more contingencies, like issues with unstowing the robotic arm or overriding excavation site preferences by the ground system. Subprocesses with

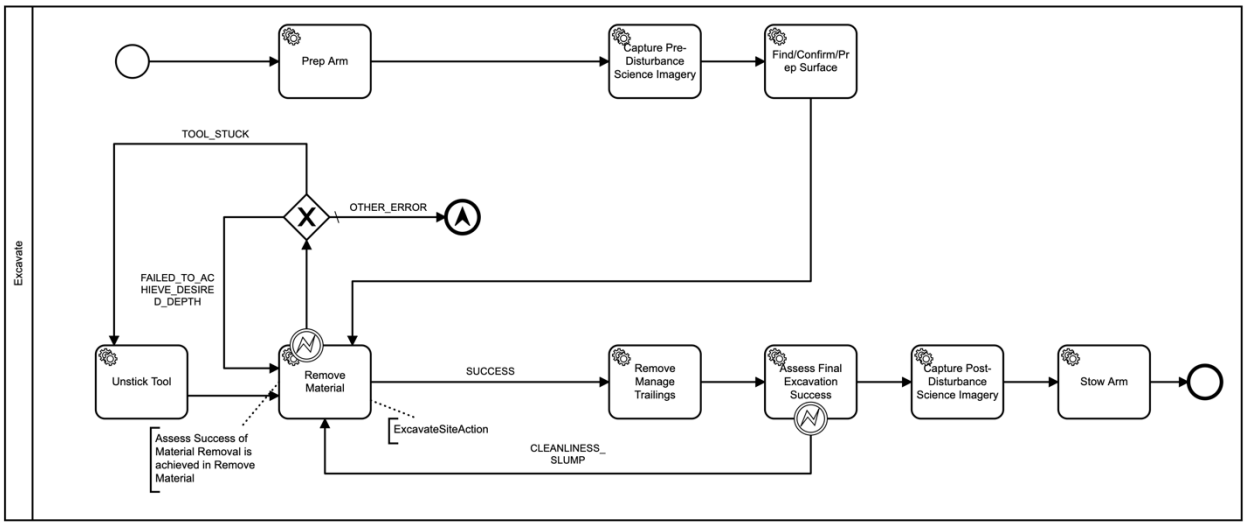

Figure 3: The Excavate subprocess in the BPMN mission model illustrating contingencies when Remove Material fails or the material slumps afterward Remove Material.

a single service task are placeholders until the actual functionality is implemented in the prototype.

In this model, the robotic activities are semantically represented by service tasks, while decisions by a ground operator would be represented by user tasks. Depending on their specification, these tasks may have a physical effect, such as moving the robotic arm, or a data effect, such as writing information into the system to use for the next activity. Upon execution, the service tasks would command the flight software (ELAP in our test case), while the user tasks would interface with ground operations. Since some of the activities are repeated for each excavation site or collection target, they are grouped in a sub-process for reuse. Within that sub-process, a boundary event on the *Excavate* call activity captures any fault generated and escalated by the excavation subprocess. As a consequence of the escalation, another site is selected by calling *Select Next Valid Excavation Option* to abandon excavation of the current site and find another option.

4.1 Fault Handling during Excavation

The nominal operation of the *Excavate* subprocess is that the robotic arm is prepared (i.e., heated to operating temperature), and then the surface is prepared for material removal. Material removal should remove enough of the surface material to reach a desired depth and then the robotic arm is stowed again. However, we could encounter a number of issues during excavation. For example, the tool used for excavation attached to the robotic arm could get stuck, the excavation may not achieve the desired depth, or the material could slump after excavation is completed. Of course, we want to be able to respond to these faults at runtime without necessarily putting the spacecraft into safe mode and awaiting ground to help rectify the issue a long while later. Consequently, the BPMN subprocess for *Excavate* illustrated in Figure 3 has additional pathways to handle the different faults identified. An error boundary event captures any faults that happen within *Remove Material* and leads to a decision gateway. If the tool is stuck, then execute *Unstick Tool* and try *Remove Material* again. If the excavation failed to reach the desired depth, then simply retry *Remove Material*. Otherwise, the issue is escalated to the parent process and this instance of *Excavate* is over. Material slumping, on the other hand, is handled after *Remove Material* completes successfully. *Assess Final Excavation Success* is another activity that uses the camera to investigate if the material has slumped after the robotic arm has finished excavating. If so, then we retry *Remove Material*. The set of responses to the fault are simple in this case, but illustrated that one can explicitly specify how the mission should flow when such a fault is encountered. An advantage to implementing these contingencies in the mission model is that it is simple to change the process flow later without having to reimplement the prototype itself.

4.2 Responding to Time and Energy States

Figure 4 illustrates the parent process in which *Excavate* is called. In this particular mission model, there is a cycle after *Select Initial Excavation Locations* that nominally selects the next best excavation site, excavates it, and then calls a *Collect All Samples from Excavation Site* subprocess to collect, transfer, process, analyze, and then communicate for each collection target within the excavation site. The executive keeps

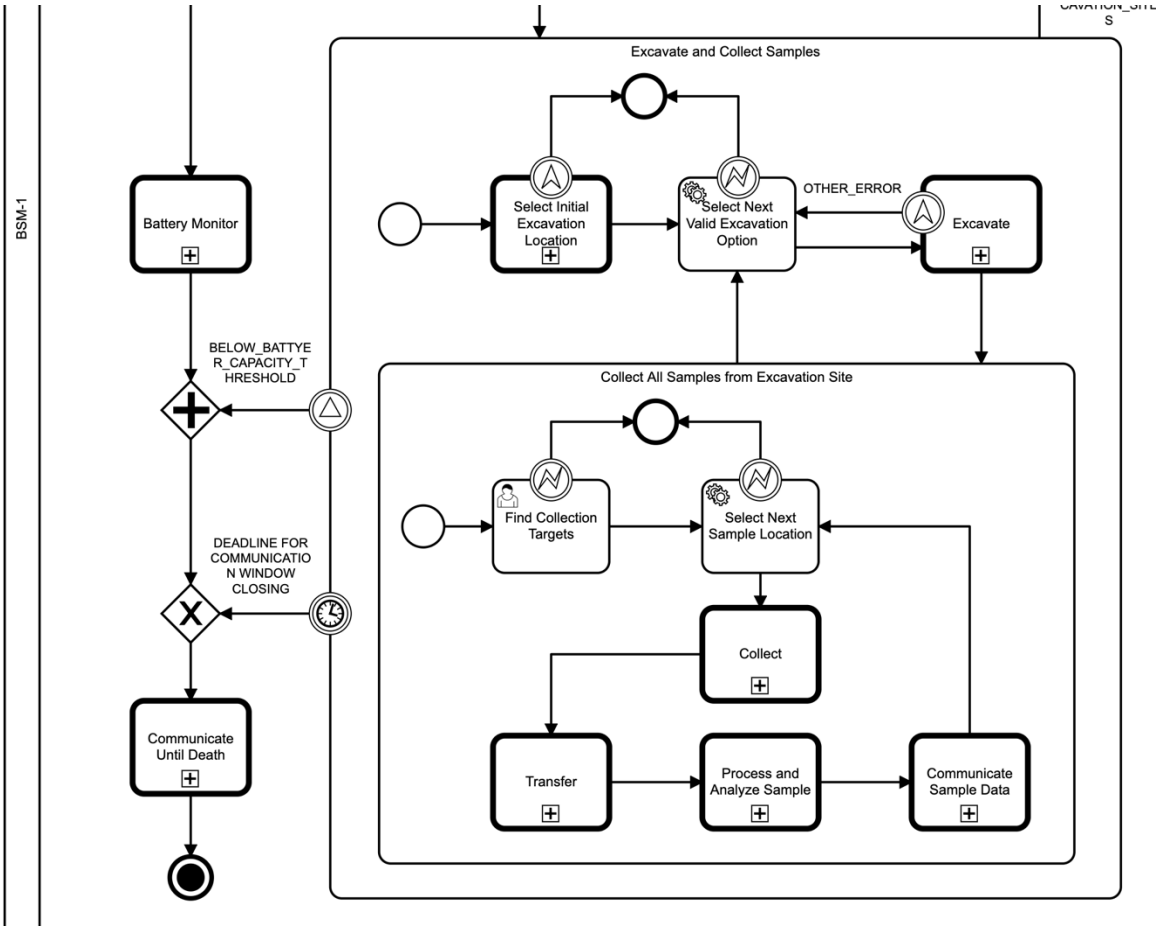

Figure 4: The Basic Surface Mission 1 (BSM-1) main process illustrating conditions when the subprocess to Excavate and Collect Samples is aborted early.

track of the current excavation site via *site_id*, which is an output of *Select Next Valid Excavation Option* and used as input to *Excavate* and *Find Collection Targets*. As previously described, if *Excavate* is unsuccessful, then *Select Next Valid Excavation Option* is called. The cycle continues until no more excavation sites are available. However, the cycle can also be aborted early due to two conditions:

1. *Time*: The *Excavate and Collect Samples* subprocess has a timer boundary event attached, which can be set to either trigger at a specific absolute mission time, or after some specified duration. Both time and duration are encoded in the ISO 8601 format. In this case, there is a deadline before the final communication window closes.
2. *Energy*: The *Battery Monitor* subprocess checks the battery state in parallel with the *Excavate and Collect Samples* subprocess. If the battery level drops below a specific threshold (e.g., 30% remaining), then a signal is sent to abort excavating and collecting samples. Alternatively, a conditional boundary event could be used to implement this constraint.

In both cases, the mission moves onto a final activity to *Communicate Until Death* and then the mission ends. Other use cases for time- and energy-based responses are, for example, responding to activities taking longer or more energy than expected or skipping activities until the communication window reopens. Additionally, thermals are another state that we are handling in an upcoming iteration of the prototype.

In addition to the general benefits of using BPMN for robotic mission modelling described in Section 2. This mission model highlights a number of other advantages:

1. *Reusability*: Sub-processes allow grouping of activities that can be invoked repeatedly like a macro. Call activities are implemented like symbolic links to some other BPMN element or process that is then invoked.

2. *Contingency Modeling*: System faults, like a collision, can be explicitly modeled and handled in any way as prescribed in the mission model. Since safe modes are integrated into the mission plan, they can return to nominal mission `flow` upon resolution or be escalated.
3. *Data-driven Execution Decisions*: Decisions can be made based on data in the system: *Are there more sample targets?* If so, then repeated sample collection with the next target.

One last note: this mission model is one way of modeling these notional Europa Lander surface operations; it is not unusual for another modeler to organize the model differently, while achieving the same mission objective.

5 CONCLUSIONS AND FUTURE WORK

TRACE is a new tool to model and execute event-driven, activity-based missions with one or more spacecraft (e.g., rover, lander, helicopter). TRACE has been previously demonstrated as an effective mission modeling and execution tool in the field. Building on this success, we have shown in this paper how TRACE and its connector (plug-in) API can be adapted to the execution of space missions involving a spacecraft and ground operations. The connector architecture was created to allow us to leverage the benefits and advantages of TRACE as an alternative (or complement) to the timeline-based mission planning and execution tools for existing autonomy subsystems. We created mission models for a representative set of robotic arm and science activities that would be executed by the Europa Lander during an autonomous surface mission. These examples were chosen deliberately to show the standardized, user-friendly, and flexible nature of the TRACE's BPMN-based mission modeling, as well as the practical aspects of using BPMN models for spacecraft missions. For example, how spacecraft missions could be modeled, how ground ops should be modeled within BPMN, how contingencies to faults are integrated into the mission model, and how data-driven execution decisions are leveraged. Future work focus on full-pipeline execution of these models simulated within ELAP, along with improvements to the verification tools, such that LTL expressions can be used to enforce resource and mission requirements prior to execution.

Appendix

Appendix includes the full BPMN mission model demonstrated with the Europa Lander autonomy prototype in June 2020 and referenced throughout this paper.

Acknowledgement

This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] V. Verma, D. Gaines, G. Rabideau, S. Schaffer and R. Joshi, "utonomous science restart for the planned Europa mission with lightweight planning and execution," Jet Propulsion Laboratory, California Institute of Technology, 2017.

[2] H. Völzer, "An Overview of BPMN 2.0 and Its Potential Use," in *BPMN 2010: Business Process Modeling Notation*, Springer, 2010, pp. 14-15.

[3] Camunda, "BPMN Modeling Reference," [Online]. Available: https://camunda.com/bpmn/reference/. [Accessed 2020].

[4] J.-P. de la Croix, G. Lim, J. Vander Hook, A. Rahmani, G. Droge, A. Xydes and C. Scrapper, "Mission modeling, planning, and execution module for teams of unmanned vehicles," in *Unmanned Systems Technology XIX*, 2017.

[5] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[6] Robotics Operating System, "actionlib," [Online]. Available: http://wiki.ros.org/actionlib. [Accessed 2020].

[7] D. Gaines and S. Chien, "M2020," [Online]. Available: https://www-aig.jpl.nasa.gov/public/projects/m2020-scheduler/. [Accessed 2020].

[8] Jet Propulsion Laboratory, California Institute of Technology, "Mission to Europa: Europa Lander," [Online]. Available: https://www.jpl.nasa.gov/missions/europa-lander/. [Accessed 2020].

[9] Jet Propulsion Laboratory, California Institute of Technology, "Europa/Ocean Worlds Lander Mission Concept," 2020. [Online]. Available: https://www.jpl.nasa.gov/missions/web/absscicon/2020_ELOW_Final_20200514_Post_v2.pdf. [Accessed 2020].

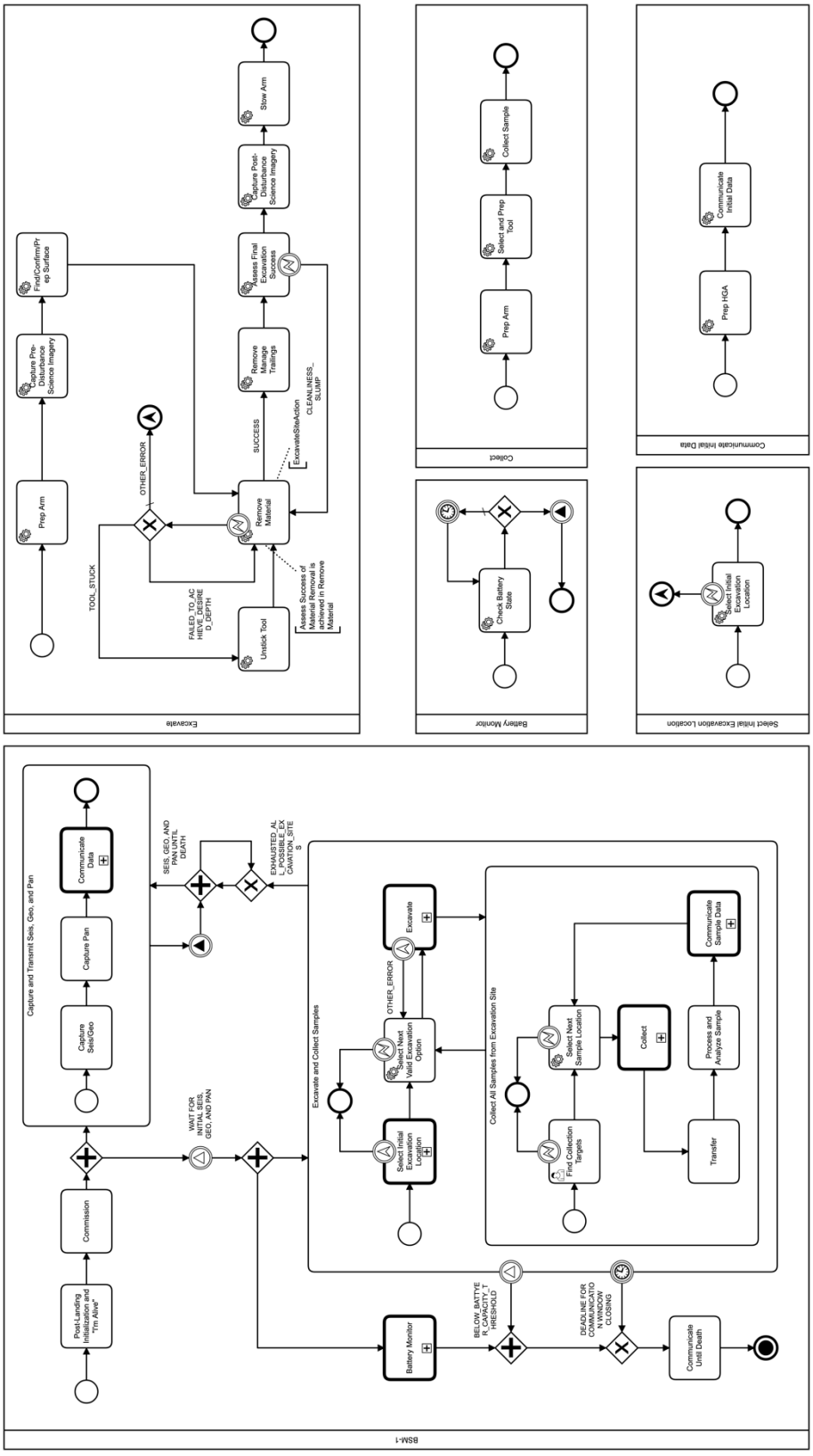

Figure 1: The notional Basic Surface Mission for the Europa Lander concept implemented in BPMN.