Requirements-Driven Configuration of Emergency Response Missions with Small Aerial Vehicles

Jane Cleland-Huang
Ankit Agrawal, Md Nafee Al Islam, Eric Tsai
JaneHuang@nd.edu
University of Notre Dame
South Bend, IN, USA

Maxime Van Speybroeck
maxime.vanspeybroeck@student.unamur.be
University of Namur
Namur, Belgium

Michael Vierhauser
michael.vierhauser@jku.at
Johannes Kepler University Linz
Linz, Austria

ABSTRACT

Unmanned Aerial Vehicles (UAVs) are increasingly used by emergency responders to support search-and-rescue operations, medical supplies delivery, fire surveillance, and many other scenarios. At the same time, researchers are investigating usage scenarios in which UAVs are imbued with a greater level of autonomy to provide automated search, surveillance, and delivery capabilities that far exceed current adoption practices. To address this emergent opportunity, we are developing a configurable, multi-user, multi-UAV system for supporting the use of semi-autonomous UAVs in diverse emergency response missions. We present a requirements-driven approach for creating a software product line (SPL) of highly configurable scenarios based on different missions. We focus on the process for eliciting and modeling a family of related use cases, constructing individual feature models, and activity diagrams for each scenario, and then merging them into an SPL. We show how the SPL will be implemented through leveraging and augmenting existing features in our DroneResponse system. We further present a configuration tool, and demonstrate its ability to generate mission-specific configurations for 20 different use case scenarios.

CCS CONCEPTS

• **Software and its engineering → Software product lines**; **Requirements analysis**.

KEYWORDS

Configuration, Product Line, Unmanned Aerial Vehicles, Emergency Response

ACM Reference Format:
Jane Cleland-Huang, Ankit Agrawal, Md Nafee Al Islam, Eric Tsai, Maxime Van Speybroeck, and Michael Vierhauser. 2020. Requirements-Driven Configuration of Emergency Response Missions with Small Aerial Vehicles. In *24th ACM International Systems and Software Product Line Conference (SPLC '20), October 19–23, 2020, MONTREAL, QC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3382025.3414950

SPLC '20, October 19–23, 2020, MONTREAL, QC, Canada

ACM ISBN 978-1-4503-7569-6/20/10…$15.00
https://doi.org/10.1145/3382025.3414950

1 INTRODUCTION

Small Unmanned Aerial, Land, or Submersible Vehicles (commonly known as drones) are increasingly deployed to support time-critical emergency response missions. In current scenarios their use is typically limited to manual operation by a remote pilot controlling a single Unmanned Aerial Vehicle (UAV); however, UAVs could be empowered to play a far more extensive role in a mission. Cohorts of semi-autonomous, self-coordinating, UAVs could collaborate closely with human first responders to support diverse activities such as search-and-rescue, surveillance, air and water sampling, medical supply delivery, and far more. For example, when used to support fire fighting efforts, a cohort of semi-autonomous UAVs could map out a building in order to generate a 3D heat map of potentially dangerous hotspots. Similarly, in an emergency search-and-rescue operation for a child swept out to sea, the UAVs could dynamically generate flight routes and use onboard image detection capabilities to coordinate an autonomous search for the victim, deliver flotation devices, and track and report the victim's location until rescuers arrive [6, 10, 32, 68].

Different missions share many common tasks but also exhibit unique characteristics. Efficiently and effectively managing such variabilities in Cyber-Physical Systems [45], such as our system of collaborating UAVs, is a non-trivial challenge that must take into consideration the context and sequencing of events, while simultaneously addressing safety-critical concerns. Tasks such as launching UAVs, obstacle and collision avoidance, or planning search routes, are common across many, or even all of the missions; while other tasks, such as collecting water samples, or tracking a moving victim in the river, are unique to a specific mission. The UAVs, including the software controlling them and the interfaces for interacting with them, must support these diverse scenarios throughout the mission [2, 22]. Variability points can be managed through inclusion or exclusion of specific capabilities, differences in the order in which tasks are executed, and through different levels of UAV autonomy and human interactions enabled across different mission contexts [11, 14, 29]. Mission-specific products must be configured and deployed quickly prior to launch and the concrete configuration pushed onto central components, mobile units, and the UAVs.

The work we present in this paper is driven by an initial set of seven community-inspired use cases for which variability points included software features, hardware capabilities, differing degrees of UAV autonomy versus human control, and alternate sequencing of tasks for different missions. Our paper makes several contributions. First, it provides *a dataset of seven detailed use cases* describing

emergency response scenarios and presents a requirements-driven process for systematically constructing an SPL feature model and activity diagram from these use cases. Second, it explores *behavioral aspects of the product line (PL)* such as sequencing of events and configurable and adaptable autonomy levels with differing degrees of human-drone interaction. Third, it *presents a practical configurator* which emergency responders can use to select predefined mission plans or to configure new mission specifications. Our solution focuses on pre-launch configuration; however, the launched product is able to adapt to changes in the environment or in response to human interactions. Our solution represents a pragmatic approach for configuring a non-trivial system which needs to execute immediately following configuration without additional testing. While all the examples in this paper have focused on the single-domain of multi-UAV emergency missions, the process we present could be useful in other CPS environments – for example, with factory floor delivery robots.

The remainder of the paper is laid out as follows. In Section 2 we provide a brief overview of our DroneResponse application and describe the use case scenarios that were foundational for developing our configurator. In Section 3 we describe the process used to create feature hierarchies and activity diagrams for individual mission scenarios and to merge them into a PL model, while Section 4 describes the actual configuration process. In Section 5 we describe our evaluation which included validation tests, a preliminary user study, and a discussion of lessons learned. Finally, Sections 6 to 8 report threats to validity, related work, and conclusions.

2 ELICITING AND SPECIFYING USE CASES

Eliciting requirements for diverse usage scenarios is a key step in creating individual feature and behavioral models from which PL assets can be constructed. In general, a PL represents a set of software-intensive systems that share a common, managed set of features developed or composed from a common set of core assets [17, 40, 51, 59, 65]. A product is typically derived by selecting a set of alternative and optional features (variabilities) and composing them on top of a set of common base features (commonalities) [5, 25, 33, 39, 41]. In this paper we focus on requirements aspects of the product line – notably the creation of the feature model and an activity diagram which ultimately controls and constrains transitions across tasks and states. The approach described in this paper is part of a longer-term project to fully implement our DroneResponse system as a configurable product line [2, 16]. We therefore first provide a brief overview of DroneResponse, and then describe the processes we have engaged in to elicit a diverse set of requirements which form the basis for developing the PL.

2.1 Drone Response System

DroneResponse is a UAV management and control system extended from our previous work on UAVs [16, 64]. Our aim in developing DroneResponse is to support diverse emergency missions from within a single application. The system will support two forms of configuration. First, emergency responders will be able to configure common missions in advance and then contextualize and parameterize them. Second, it will enable configurations of new, previously unseen and unfolding scenarios prior to launch, based on allowed combinations of features and sequences of behavior.

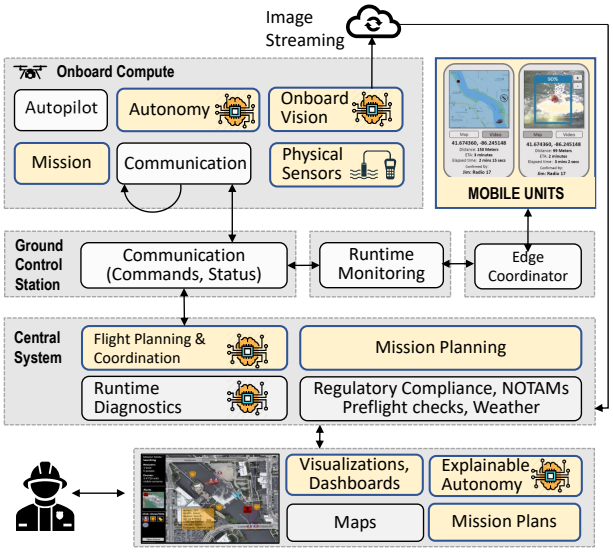

Figure 1: DroneResponse high-level architecture overview. The yellow components will be configured for each mission.

Fig. 1, depicts the high-level architecture of the DroneResponse system and shows some of its variation points. All UAVs are imbued with basic decision-making capabilities driven by the general mission goals (e.g., find and rescue a victim in the river), constrained by UAV capabilities (e.g., thermal vs. visible light camera, or flotation device delivery capabilities), and influenced by their assigned roles and responsibilities. UAVs leverage their knowledge of the environment and the overall current state of the mission in order to make decisions on enacting specific tasks. They share information with other UAVs as well as centralized modules that assist with global coordination and meta-reasoning tasks.

2.2 Requirements Discovery

We used three distinct sources of information to identify requirements for diverse mission scenarios. We started with the requirements, features, and architecture of our existing Dronology platform [16] which was created and developed through close collaboration with the South Bend Fire Department [2]. Next, we searched for grey literature to find descriptions of emergency responders, firefighters, coast guards, and other groups using UAVs in emergency response scenarios. Here our goal was not to retrieve a complete or exhaustive set of use cases, but to select a set of well-described and diverse scenarios for driving the development of our PL. The references we ultimately used are listed in Table 1. Each of the described scenarios either involves UAVs being controlled manually by a human operator or the use of an existing off-the-shelf solution such as MissionPlanner, QGroundControl, or DroneSense[1], to plot or generate sets of waypoints for the UAVs to follow. As a final step, we performed an informal second literature search, this time for academic papers describing the use of autonomous and/or semi-autonomous UAVs for emergency response missions.

[1]URLs: ardupilot.org/planner, qgroundcontrol.com, www.dronesense.com

Use Case: Ice Search and Rescue
ID: SPLC-11

Description
UAV(s) dispatched with a flotation device for ice rescue

Primary Actor
Drone Commander

Trigger
The Drone Commander activates the delivery.

Main Success Scenario
1. Emergency responders **plan_area_search** [SPLC-1001]
2. The DroneResponse commander issues a command to start the mission.
3. The UAV(s) **takeoff** [SPLC-1007]
4. The UAVs **perform_search** [SPLC-1002]
5. The UAV **requests_victim_confirmation** [SPLC-1005] from the human operator.
6. The UAV receives confirmation from the human operator that the victim sighting is valid.
7. DroneResponse automatically sends the GPS coordinates to the mobile_rescue system.
8. The UAV switches to **flotation_device_delivery** [SPLC-1006] mode.
9. Human responders reach the victim's location and execute a rescue.
10. The Drone Commander **ends_mission** [SPLC-1007].

Specific Exceptions
1. In step 3, one of the UAVs fails to take-off.
   1.1 If a replacement UAV is flight-ready, it is dispatched in place of the failed UAV.
   1.2 If no replacement is available DroneResponse re-executes **generate-search-plan** [SPLC-1009] for the available UAVs and previously defined search area.

General Exceptions
1. At any time, if communication is lost between the Ground Control Station and a UAV, DroneResponse executes the **Lost Drone-to-GCS Communication** (SPLC-2001) exception case.
2. At any time, a malfunction error is raised by a UAV in flight, DroneResponse executes the **Drone-in-flight Malfunction** (SPLC-2002) exception case.

Figure 2: A partial view of the "Ice search-and-rescue" use case focusing on the main success scenario and examples of general and specific exceptions. Requirements were derived from news reports and video footage (e.g., [1, 56])

One researcher from our team conducted the preliminary search, using the search term "UAV & Emergency Response", followed by a snowballing process to retrieve papers referenced by the initial search. Our goal for this second literature search was to identify cutting-edge solutions for UAV autonomy and their applications to emergency response. Relevant papers are again listed in Table 1 under the 'Acad.' (academic research papers) column[2].

2.3 A Use Case Driven Approach

Textual use cases document requirements from the perspective of their external actors by describing a primary sequence of actions, as well as alternatives and exception cases [18]. They are therefore able to capture both mandatory and optional features [26]. Bertolino *et al.* [7] extended the standard use case notation to support product lines by using constraints to document legal versus illegal combinations of variants; however, this approach leads to early formalization of the product line which we found detrimental to the task of exploring and documenting stakeholders' needs. We therefore adopted a less formal and more flexible approach to specifying use cases [18]. This enabled us to focus on eliciting and understanding the requirements for individual mission scenarios without prematurely focusing on the more complex challenge of building a PL. As a result, we deferred the identification of commonalities and variabilities across scenarios until later in the process.

An example use case for UAV deployment in an "Ice search-and-rescue" scenario is depicted in Fig. 2. The use case starts by describing the actors and stakeholders and establishing pre-conditions and post-conditions. It then describes the *main success scenario*, as well as *alternate sequences of events* and *exceptions* that can occur at any time of the mission. Steps that describe common tasks, shared across multiple use cases, are defined as references to supporting use cases, while steps that are specific to the ice-rescue use case are described directly in the text. Due to space limitations the use case shows only the main scenario steps, with a few examples of specific- and general exceptions.

Table 1: Scenarios were derived from real-life accounts, stakeholders, and academic literature.

ID	Use Cases	Usage	Acad.	Contrib. Stakeholders
UC1	River Search & Rescue	[2, 15]	[61]	South Bend Firefighters
UC2	Ice Rescue	[1, 56]		News reports
UC3	Defibrillator Delivery	[13, 47]	[28]	DeLive, Cardiac Science
UC4	Traffic Accidents	[48, 53]	[42]	South Bend Firefighters
UC5	Structural Fires	[34]		South Bend Firefighters
UC6	Water Sampling	[46, 50]	[43]	Environmental Scientists
UC7	Air Sampling	[12, 57]	[3, 69]	Environmental Scientists

3 REQUIREMENTS-LEVEL PL MODEL

Our goal is to support two distinct types of configurations. In the first case, the SPL should facilitate the configuration of known mission scenarios. Initially these will include river search-and-rescue, ice rescue, defibrillator delivery, traffic accident monitoring, structural fire support, and water and air sampling (see Table 1). In the second case, the SPL should configure previously unseen mission scenarios through the reuse of existing features combined and sequenced in new ways. Our process involves the two primary steps of modeling and configuration as illustrated in Fig. 3.

3.1 Modeling the PL

Product lines are characterized by their commonality and variability points – all of which need to be concisely modeled. As our focus in this paper is on the Requirements Engineering phase of a SPL, we focus on generating a feature model that captures commonalities, variability points, and feature constraints, and an activity diagram that describes dynamic aspects of the system by modeling the flow from one activity to another [4].

3.2 Requirements Modeling

Our requirements modeling process involves the following steps:

Step M1 – Specify Use Cases: The requirements elicitation process was performed in an iterative fashion [31], starting with the

[2]Use cases are available at https://github.com/SAREC-Lab/sUAS-UseCases and will be updated and extended as part of our ongoing work.

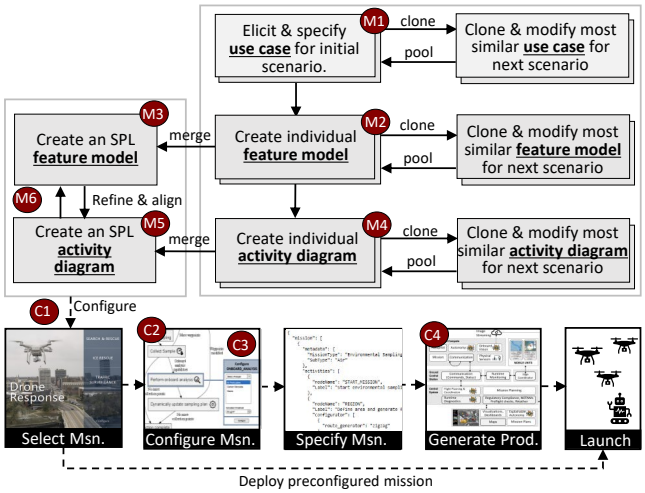

Figure 3: We followed an iterative approach to create individual use cases, feature models, and activity diagrams, and then merged them into PL-level models used with our product configurator (See M1-M6). Individual missions are then configured from the PL-level models (See C1-C4).

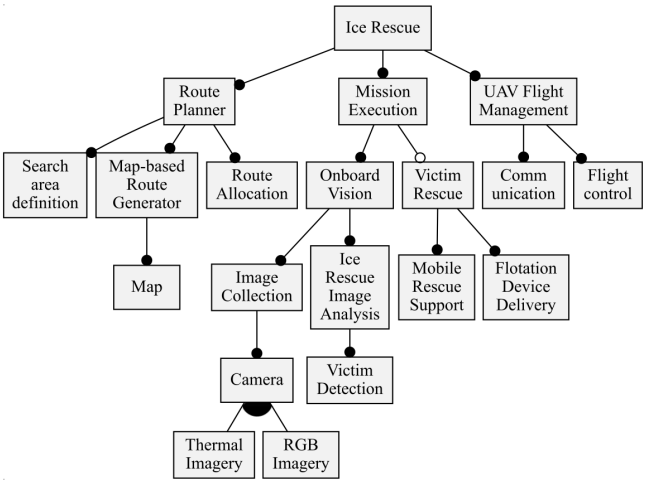

Figure 4: A mission-specific FM for the Ice rescue scenario in which UAVs support emergency responders in rescuing a person who has fallen through the ice.

"River search-and-rescue" use case from our existing system specifications [2, 15]. We cloned this original use case and derived a use case for a second mission scenario (Ice Rescue) by adding, removing, and modifying use case steps [27]. While this type of "clone-and-own" approach is commonly used in industry to design and develop complete products including their requirements, architecture, code, and tests; we applied it only to the upfront requirements elicitation and analysis phase. The clone-and-own approach of writing use cases for each scenario, has the advantage that semantically similar use case steps and sequences of activities are shared across use cases, enabling semi-automated support for the SPL construction process. We followed this process to create a use case for each of the mission scenarios shown in Table 1. For each use case we identified and cloned the most similar existing use case as the starting point.

Step M2 – Construct Mission-Specific FM: For each mission scenario we then created an individual FM. Starting with the "River search-and-rescue" (UC1) scenario, we manually identified the features needed to support the use case. We composed them into a hierarchy of mandatory, optional, and alternative features. Where necessary we added additional cross-tree constraints; however, these were rare within each of the individual mission models. The resulting FM was added to an FM pool. For each subsequent mission, we selected the most similar FM from the pool, cloned it, and then manually adapted it for the new mission by adding, removing, and modifying features, associations, and constraints. The feature hierarchy for Ice Rescue is depicted in Fig. 4. Each FM was specified in a JSON format; however other formats such as XML would also have been suitable. Each node was assigned a *NodeName* and a descriptive label. Associations between features were specified as parent-child relationships between nodes, and constraints were structured as tuples (NodeName, NodeName,[requires|excludes]).

Step M3 – Semi-automated Merge of Individual FMs: In the next step of the process, we merged all of the individual FMs into a single PL-level FM. We followed an incremental approach as this limited the complexity of reconciling differences at each individual step. We started with the River search-and-rescue FM, treating it as the initial baseline, and used a simple automated name matching algorithm to automatically merge the "next" individual FM into the current baseline. Given the clone-and-own approach in which many *NodeNames* were shared across individual FMs, the majority of nodes were automatically matched using a simple *NodeName* matching algorithm. After each merge, we manually inspected and refined the resulting model to correct any problems through (1) merging nodes that served a similar purpose but had failed to merge due to having different node names, (2) reconciling different hierarchical organizations of the same features, and (3) standardizing the level of refinement across FMs by refining leaf features that were at higher levels of abstraction than defined by other FMs. On average each merge required approximately 5-15 minutes of effort depending upon the delta between the two models. The philosophy for selecting the "next" model to merge, was diametrically opposite to the philosophy of selecting a use case or model to clone. Instead of selecting the most similar model, here we selected the most dissimilar FM to the currently merged baseline. This meant that we were able to address major structural differences in earlier stages of the merging process before the model grew in size and complexity. The merged feature model is depicted in Fig. 5.

Step M4 – Construct Mission-Specific Activity Diagrams: Each of our missions is characterized by a carefully choreographed set of human activities and semi-autonomous UAV tasks. For example, during a search-and-rescue mission, humans and UAVs first plan the mission, then engage in a search activity, and when the victim is found, they transition to activities such as tracking or delivering a flotation device. In some missions, a special UAV equipped with a flotation device might be waiting on the sidelines for the victim to

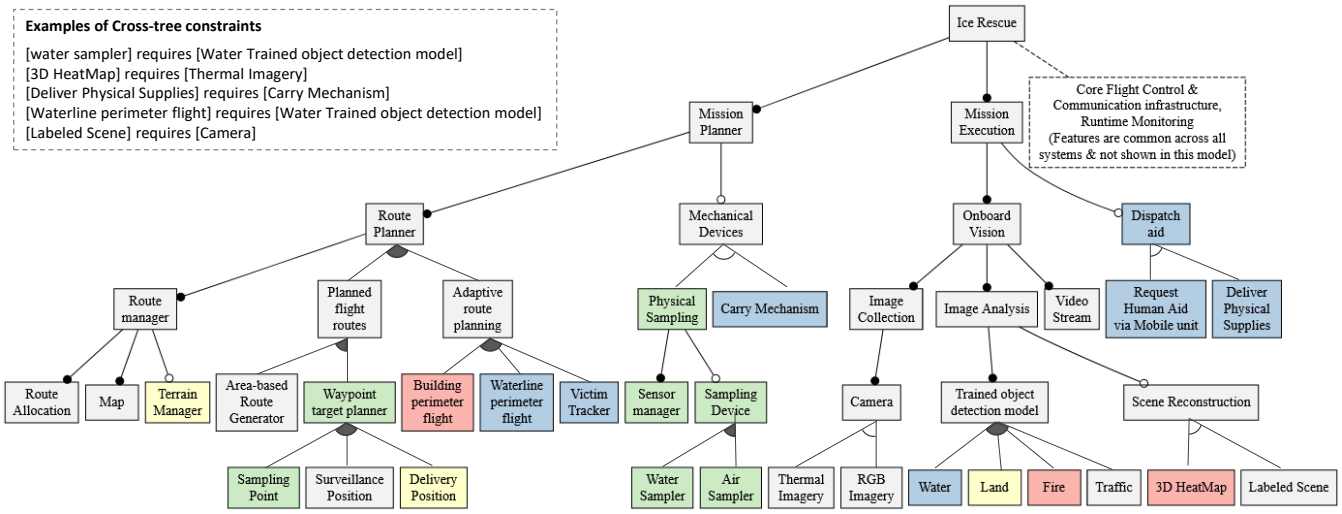

Figure 5: The merged Feature Model includes features from river and ice rescue (blue), delivery (yellow), structural fires (red), environmental sampling (green), and traffic surveillance. Features used in multiple scenarios are shown in gray. Cross-tree constraints are primarily "requires" dependencies.

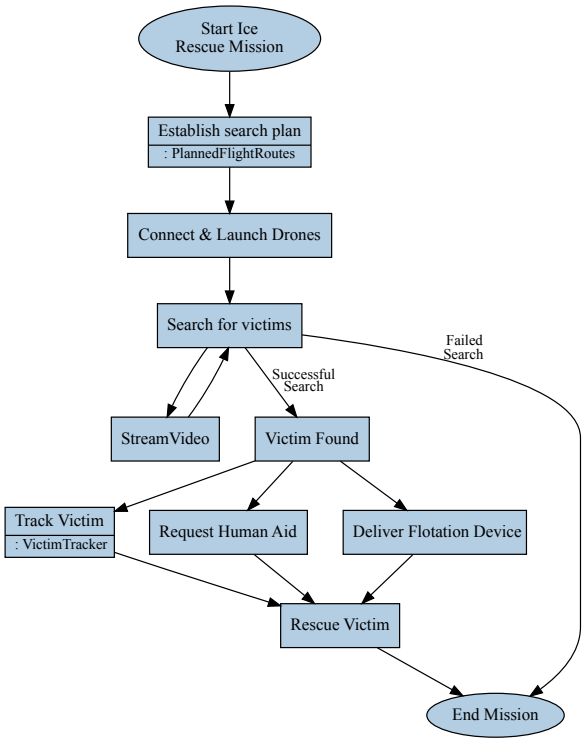

Figure 6: Activity diagram showing the sequencing of tasks for the Ice Rescue scenario. By default, transitions occur when the current task is complete. Some transitions (e.g., "Failed Search") are explicitly labeled.

be found, while in another mission (perhaps search-and-rescue at sea), all UAVs might be equipped and dispatched with life-saving devices from the start of the mission. The sequencing of activities is therefore different for each mission, and is documented in the form of an activity diagram. An example for Ice-Rescue is shown in Fig. 6. Activity diagrams not only document activities and their transitions, but are used for two additional purposes. First, they are used during the product configuration process to communicate the emerging mission steps to users. Second, (although not yet implemented) we plan to use them to dynamically visualize the current state of the mission by showing the current task that each UAV is performing at any time.

Constructing a mission-specific activity diagram is again performed manually for each use case using the same cloning approach used to create the individual FMs. We experimented with different levels of model granularity, and found that visualizing detailed activities obscured the main purpose of the mission, while overly high-level abstractions hid important information about the configuration. To balance these competing needs, we modeled variability points (e.g., track victim, deliver flotation device) that had a major impact on the sequencing of the mission, whilst hiding internal configuration points such as computer vision models or route planning algorithms used for specific activities within specific contexts. We also hid sequencing variations which impacted a single higher-level task. For example, the track-victim task involved a variant that was driven by the autonomy level of the UAV. A UAV awarded high levels of autonomy would switch automatically into tracking mode when it detected a potential victim, while a UAV with lower levels of autonomy would seek human permission before transitioning modes. This is also an example of a runtime configuration point as a user can modify the UAV's autonomy levels.

Step M5 – Semi-automated merge of individual activity diagrams: Given a set of individual activity diagrams, we merged them into a single SPL-level activity diagram following a similar process used for creating the SPL FM from individual FMs. Once again, we incrementally merged the "next" activity diagram into the baseline using a strict *NodeName* matching approach, and then

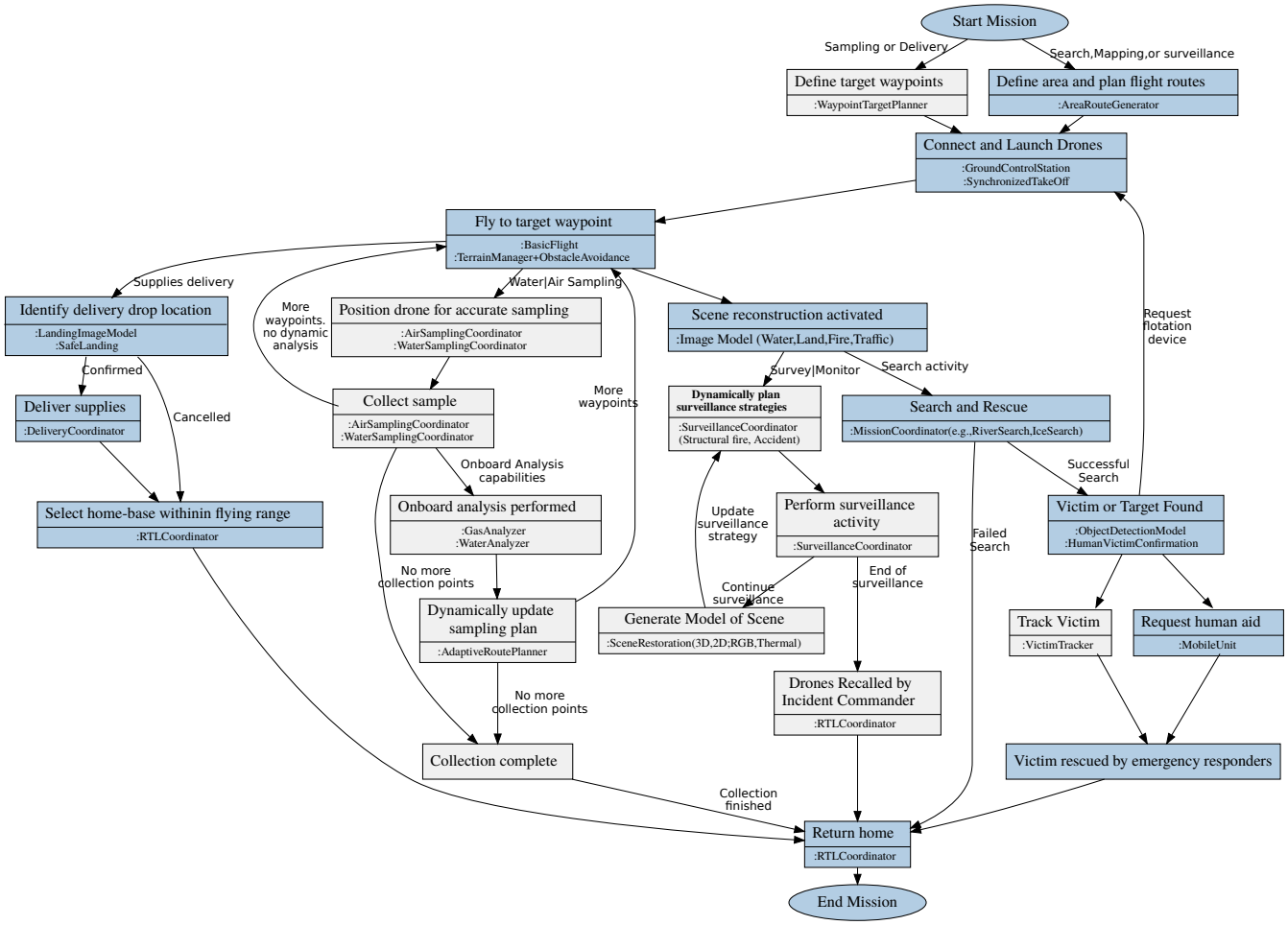

Figure 7: Merged activity flow diagram for the seven initial DroneResponse missions. Mappings to features are shown for key activities. For illustrative purposes, all activities relevant to the "Ice Rescue" scenario are highlighted in blue.

systematically refined the resulting diagram to create a new baseline. The merging process involved (1) combining activity nodes that served a similar purpose but had different node names, (2) analyzing different sequences between tasks and either reconciling them into a common sequence or establishing conditions on each transition to control the order of execution, (3) standardizing the level of refinement across diagrams by combining low-level activities into more abstract ones, and (4) where alternate flows of specific activities were identified for different missions, abstracting the activity to a higher level and making that activity node configurable according to the mission context.

Step M6 – Reconciling Models and Mapping Tasks to Features: In this final modeling step, we first manually created a mapping from each feature in the FM to concrete components (either existing or planned) in our current implementation. We were able to create mappings to approximately 25% of the variability points and almost 75% of the core flight control mechanisms and runtime monitoring components. Where components did not yet exist, we mapped to components that are *planned*. Second, we mapped the activity nodes to features. As *NodeNames* in the FM and activity diagrams did not tend to match, we also performed this step manually. We show the most important components in the lower half of each activity node in our PL activity diagram (see Fig. 7). These mappings will be used to physically configure the product for a defined mission. This step is critical for reconciling the FM and activity models, and for connecting them to the concrete system architecture.

4 MISSION CONFIGURATION

We used the PL-level FM, activity diagram, and their mappings to the concrete implementation to develop a product configurator for deriving new missions. Deelstra *et al.* [20] highlighted several product derivation challenges experienced in industry, including high reliance upon experts, incompatible component interactions, and erroneous parameter settings. Each of these represents a non-trivial problem which must be effectively addressed in the DroneResponse environment where the goal is to configure and immediately execute the mission. Improper configurations could cause serious failures in the mission, such as failure to dispatch a UAV with a flotation device when needed, failure to identify a victim due to

Table 2: Questions asked during the configuration process

ID	Question	Variants
Q1	What type of mission?	Fire, Environmental sampling, Search, Delivery, Surveillance
Q2	How will you define flight paths?	Region, Waypoints
Q3	Are you fighting a structural fire or a ground fire?	Structural, Ground
Q4	What type of environment are you working in?	Water, Land, Ice, Snow
Q5	What are you surveying?	Traffic, Flood, Other
Q6	Are there independent rescue teams?	Mobile rescue unit
Q7	Should drones track the victim?	Victim tracking
Q8	Will your mission deliver rescue equipment to the victim?	Rescue equipment
Q9	Do drones have onboard sample analysis capabilities?	Onboard analysis
Q10	What are you sampling?	Water, Air

incorrect vision settings or models, or a communication failure that results in a mid-air collision. Furthermore, our configurator will be operated by emergency responders under time-pressure to respond to the emergency situation, hence there will be no opportunity to inspect, review, or validate new configurations before flight.

All of our missions share the same architecture, populated with mission-relevant components such as computer vision models, analytics, UIs, and autonomy models onboard the UAVs. While the FM offers a great deal of flexibility in the way components can be combined into a new product, we currently limit configurations to those explicitly supported by our PL activity diagram. This reduces risks of unexpected, unsafe, previously untested feature interactions inherent to our configure-and-launch environment, while still allowing a significant degree of flexibility due to internal configurations of each activity node. We will relax these constraints in future phases of our project.

4.1 The DroneResponse Configuration Process

DroneResponse missions are derived through a series of four configuration steps (C1-C4), highlighted in Fig. 3, that impact the central server, ground control stations, onboard compute, and interconnected mobile devices. The configuration process dynamically generates a visualization of the configured mission (cf. Fig. 8) which is primarily used for communicating with the user and for supporting interactive configuration of activity nodes. It also outputs a machine-readable mission specification in a JSON format which is passed to the back-end server and used to manage product configuration and deployment.

Step C1 – Assembly/Configuration Choice: The user either selects an existing mission to configure, or chooses to assemble a new mission. If the user selects an existing mission, they can either launch the mission as-is (bypassing all remaining configuration steps) or configure the existing mission (bypassing step C2).

Step C2 – High-Level Mission Assembly: DroneResponse provides a wizard to guide the user through the process of assembling a novel mission. We identified 10 initial questions, shown in Table 2, which were needed to differentiate the primary goals and contexts of the seven use cases listed in Table 1. Some questions have several candidate answers, listed as variants in the table, while others required yes/no answers to specify whether a feature was present or absent. The questions are organized into a decision tree so that pertinent follow-up questions can be asked in response to previous answers. For example, if a user responds "SEARCH" to question Q1, then they are asked questions Q4, Q6, Q7, and Q8 in order to determine planning, context, rescue, and tracking capabilities. The maximum number of questions per configuration is currently five, and the least number is two. Once these questions are answered, DroneResponse generates a mission-specific activity diagram in Graphviz format which is dynamically rendered on the screen. Fig. 8 shows the activity diagram rendered for an environmental air sampling scenario.

Step C3 – Component Configuration: We annotate configurable nodes with representative icons as depicted in Fig. 8, which shows five configurable nodes. For example, area definition and flight route generation features can be configured with different mapping options (terrain, map) and with different techniques for generating flight routes. Similarly sample collection and onboard analysis activities are configurable by available collection mechanisms and onboard analysis techniques, and software for connecting to mechanical controls (sensors and actuators) and for performing onboard analysis needs to be correctly loaded onto the UAVs prior to the mission.

Step C4 – Runtime Configuration: Finally, we support a limited number of runtime configuration options. Some of these are exposed to users while others are not. For example, the synchronized launch mechanism is automatically activated at runtime if more than one drone is scheduled for simultaneous launch, and we do not allow the user to turn off this feature. However, other cases are exposed to the users – for example, allowing them to raise or lower the autonomy permissions of the UAVs with respect to specific actions. A specific example, was discussed earlier with respect to victim tracking, as a user could reduce the UAV's autonomy and require that it seeks permission to track a candidate victim.

4.2 Deployable Configuration

This multi-step configuration process produces a mission specification in JSON which serves as a machine-interpretable description of the mission and its configuration points. Configuration decisions impact the system in several key ways including the following:

- **Central Control Mechanisms:** Some parts of DroneResponse are centrally coordinated. Examples include the *route_planning* algorithms that generate search routes for N UAVs within a region defined by the user. Core Dronology components are configured dynamically through a parameterization mechanism.
- **Onboard Autonomy:** DroneResponse UAVs are imbued with decision-making capabilities using the *BDI* (Belief-Desire-Intent) model [55] of autonomous agents. UAVs build their beliefs through directly observing the environment (e.g., use of their onboard sensors) and through receiving information from other UAVs, central control, and human operators [67]. Mechanisms for supporting communication and enabling UAVs to reason about achieving their goals (desires) through enacting specific tasks (intents) are

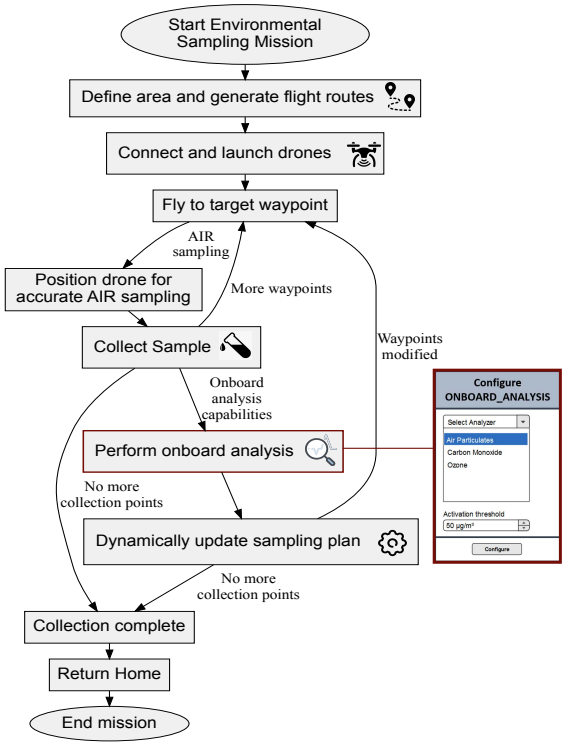

Figure 8: The mission is configured and visualized according to user responses. Icons represent configurable nodes.

included in the mandatory architectural components [52] on board each UAV. Onboard configuration requires configuration of several BDI components, including its knowledge management capabilities, goals, permitted tasks, and mode transitions. Onboard configuration also involves context-specific computer vision models, logic for interacting with mechanical devices, and software analytics for dealing with specialized data (e.g., onboard water sampling).

- **Mobile Units:** The DroneResponse system represents a complex socio-technical environment that includes mobile units which also may need to be configured or activated. For example, our current River Rescue system includes a mobile application for communicating with rescuers on a boat.
- **User Interface:** Missions can be configured to support different degrees of human-drone interactions [14] within mission-specific contexts. Therefore UIs need to be activated and configured according to the specific mission. An example of a mission-specific UI is shown in Fig. 9.

The mission coordinator component on the central DroneResponse system is responsible for interpreting the mission specification, configuring each part of the system, and using the mission specification to track and monitor the state of the unfolding mission. However, in this phase of the project we have focused on the upfront requirements elicitation and the mission specification phase of the configuration process. Using the resulting mission specification to configure DroneResponse is therefore outside the scope of this paper.

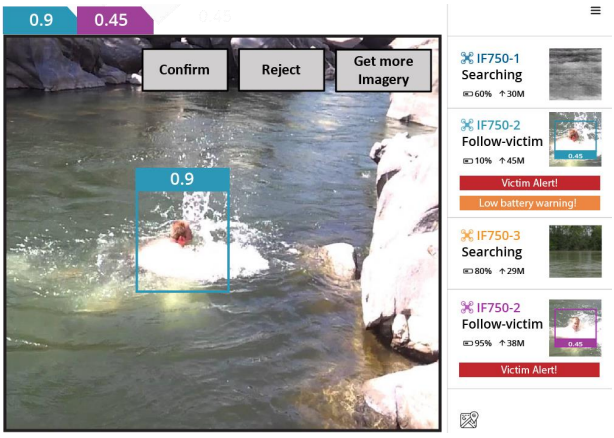

Figure 9: The DroneResponse UI allows an operator to review and evaluate video footage of a potential victim.

Table 3: Additional use case scenarios used for validation tests and for the user study. Participants (P1-P5) were assigned two use cases, and selected a third one of their choice.

ID	Use Case Name		ID	Use Case Name	
UC8	Chemical spill	P1	UC15	Flood support	P4
UC9	Avalanche rescue	P1	UC16	Earthquake damage	P5
UC10	Suspect tracking	P2,P3	UC17	Rip current rescue	P5
UC11	School shooting	P2,P4	UC18	Lost kayaker	P2
UC12	Radiation detection	P3	UC19	Volcanic eruption	P5
UC13	Man overboard	P3	UC20	Utility inspection	
UC14	Crowd control	P1,P4			

4.3 Implementation

DroneResponse's UI is designed specifically to support emergency response missions [2] and is developed using the Angular Framework. We have implemented and integrated the configurator into the DroneResponse UI so that users can interactively answer the configuration questions. The resulting mission specification is dynamically visualized as an activity diagram using the services of the d3-graphviz library. In addition a machine-readable JSON file is generated depicting the mission configuration.

5 EVALUATION AND ANALYSIS

We evaluated the modeling and configuration process and the resulting configurator by addressing the following research questions:

- **RQ1:** Is the configurator able to generate a valid mission specification for the seven primary use cases that were used in the SPL design and also for a set of previously unseen use cases?
- **RQ2:** Are users able to use the configurator to generate a valid mission specification for a diverse set of use cases? If not, what challenges exist?
- **RQ3:** What challenges were identified when using the proposed process to create an PL FM and activity diagram?

RQ1: Validating the configurator

To address RQ1, three researchers on our team conducted a series of *validation tests* using the configurator to generate products for each of the original seven use case scenarios (see Fig. 1). We then performed walk-throughs of the visualized missions (e.g., Fig. 8) to evaluate them against their original use cases. Following minor corrections and adaptations (e.g., a missing link), all scenarios passed their test cases and produced the intended mission specification.

Next, we repeated the validation using the additional set of use cases shown in Table 3. These use cases were specified less formally than those used to build our PL, and were written by our team based on additional examples we found for UAV deployment in emergency response scenarios. The tests revealed a few issues. For example, in UC11 (School shooting), available configuration options for search-and-rescue adequately covered the basic use case of finding the suspect except for terminology mismatches such as the use of "victims" in place of "suspects". The problem could be solved by inserting an additional question (e.g., "Is this a law enforcement activity?") to establish the correct terminology, or by choosing a more generic term (e.g., target). We adopted the former option for our user study.

RQ2: User Study of the Configurator

To address RQ2 we recruited five participants, as that is the number recommended by Nielsen for effective and efficient early-phase design evaluations [49]. All of our participants had experience flying UAVs (two with FAA Part 107 remote pilot licenses, and three as hobbyists). Three had experience working in scenarios with multiple physical UAVs, and one in a simulated environment with multiple UAVs. One of the certified pilots had no experience in multi-UAV environments. The other certified pilot had previously worked directly with the Fire Department in a UAV training exercise. Each session was supervised by two of the authors and conducted over Zoom. It took approximately 30 minutes. For training purposes, all participants viewed a short video describing the DroneResponse project [3]. The researchers then demonstrated the use of the configurator for UC1 (River search-and-rescue), thereby simulating the training we would expect users to have prior to using the configurator.

The participants were each asked to configure three missions. Two of the missions were preassigned as depicted in Table 3, while they were allowed to choose the third scenario from the complete list. We recorded their actions and audio using Zoom. Furthermore, we asked participants to follow a "think-aloud" protocol [23] as they configured the mission. At the end of each session we asked the users three questions: Q1: "Did the configuration wizard ask the right questions to help you configure the mission?'', Q2: "Does the visualization help you understand the current mission configuration?", and finally Q3: "Do you have any suggestions for improvements?'.

We transcribed the recordings, analyzed the responses, and used an inductive coding approach [62] to tag each comment. We then grouped similar tags in order to identify recurring themes associated with terminology, question flows, appearance of the visualized mission, and configuration techniques. In response to Q1, two respondents stated that the wizard asked the correct questions with comments such as "The workflow is very helpful." and it "Gets an idea what the mission can be", while three made suggestions to add more detailed questions such as asking for information about what the UAV would be carrying. All five participants agreed that the visualized workflow was very helpful (Q2). The participants also suggested improvements (Q3). One observed that it would be very important to translate all terminology into the first-responders' language (e.g., replacing "scene reconstruction" with "visualize the scene"). One suggested coloring the nodes that were generated each time they answered a question, so that it was easier to follow new parts of the mission as they were visualized. Two participants asked whether they could immediately configure the search area. As the search area will be defined after the product is configured for a mission, we have removed it from the configurator.

In general, the user study provided important confirmation that our approach for configuring the mission dynamically was well received. We plan to integrate feedback from this study into the next design iteration, which will be used in a field-study with firefighters and physical UAVs in our outdoor testing environment.

RQ3: Analysis and Lessons Learned

To address RQ3, we reflected on the lessons we learned from applying our process to the UAV-Emergency response domain and made the following observations.

- **Reliance on informal use cases?** Most automated, or semi-automated techniques that aim to extract FMs require some degree of formalization – for example using formal use cases [7, 30] or formally specified feature models. We opted to directly construct a feature hierarchy for each use case scenario as this provided a visual and intuitive environment for exploring mandatory and optional features as well as any constraints that existed. Our use of clone-and-own resulted in a significant number of common node names across individual FMs, which allowed for a high-degree of automation during the model merging without sacrificing details of each scenario. Keeping use cases less formal further allowed us to leverage their inherent strengths during the requirements elicitation process.
- **Constraining the Feature Model by the Activity Diagram?** Our choice to constrain the product configuration to the predefined sequences allowed by our activity diagram was a pragmatic one given the configure-and-fly nature of our product. While unwanted feature interactions can still be introduced through untested combinations of configurations across activities, we can reduce this risk through activity-level testing. All permitted transitions will be tested prior to mission execution. In ongoing work we are investigating an alternate approach that allows greater freedom in the configuration process, but utilizes temporal logic to constrain and check for valid execution sequences [44, 58].
- **Incremental vs. Big Bang Merging?** Initially we planned to merge all of the individual FMs together at the same time; however, we found the complexity of dealing with inconsistencies in node names and structure from multiple FMs was difficult to handle. The incremental approach of merging FMs and Activity diagrams one at a time into the PL models reduced this complexity significantly. Furthermore, selecting the most dissimilar model to merge into the baseline, allowed us to address any major

[3] https://youtu.be/x1pl8alCEeM

discrepancies earlier in the process while the models were still relatively simple.
- **Granularity of Activities?** We explored various granularity levels for activity diagrams, but ultimately adopted a relatively coarse-grained approach that matched the variability points of different missions. Our decision was strongly motivated by the rationale that activity diagrams will be used to communicate the mission specification to emergency responders on the scene of an emergency response and need to be quick and simple to understand. We complemented this decision by allowing configuration within individual nodes.

6 THREATS TO VALIDITY

There are several threats to validity for our approach. First and foremost, we have focused on the upfront process of eliciting requirements, engaging domain experts in the configuration process, and ultimately generating the mission specification. In our analysis we have focused on specific functional and non-functional requirements relevant for the "main success scenario" of the different use cases. Further investigation is required on how other types of requirements such as relaxable [66] or awareness [60] requirements can be included and represented in the feature model. We have provided only a high-level explanation of how our DroneResponse architecture will use the specification to execute diverse missions. This specification is the input to our next-phase task of refactoring DroneResponse as a PL; however, the engineering process may reveal changes needed in the specification as a result of a typical design process. Second, our study participants were UAV-flyers but not emergency-responders, and our planned (post covid-19) study with emergency responders [2] will inevitably introduce additional feedback requiring further evolution of our configurator and the underlying architecture. Finally, while we have applied our approach to the domain of UAVs, we envision that the process could be used across a broader range of multi-agent systems, for example in developing product lines of factory-floor robots; however, we have not yet validated this supposition. Our future work will therefore explore more diverse applications.

7 RELATED WORK

Related work includes multi-agent task specification, use cases for variability management, mapping of their variability points to components or features in PLs, and product derivation.

Multi-Agent Task Specification: Recent work by Garcia *et al.* [30] presented PROMISE, a domain-specific language for specifying missions and high-level goals for autonomous robots. Similar work has used task specification trees [21] and UML statecharts [63]. These approaches provide a strong formal foundation, which could be applied as an underlying layer in our project. However, we focused on use cases and their corresponding activity diagrams as they are conceptually easy to understand for emergency responders who will use and configure DroneResponse in the field.

Use Cases and Variability: Halmans and Pohl [38, 51] presented a requirements-driven approach for describing the variability of an SPL. They employed use case diagrams to capture requirements and communicate essential product family variability to the customers. Bühne *et al.* [9] also described a scenario-based approach based on an orthogonal variability model to support requirements engineers across all phases of the requirements engineering process for an SPL. In contrast, we use a textual description of use cases and ultimately merge these in a step-wise process to create a variability model that serves as the basis for deriving and configuring a specific scenario at runtime. The PLUSS approach [24] by Eriksson *et al.* leveraged use case scenarios and use case realizations. Unlike our approach, they maintained a single use case model for the entire product line by relating one or more scenarios with a feature in the FM. While this supports the definition of variants within the use case specification it provides inadequate support for other forms of variability such as computer vision models or algorithmic solutions that are needed to configure a mission.

Feature Model Mappings: Braganca and Machado [8] introduced an automated mapping between use case diagrams and FMs using a model-driven approach to explore relationships between UML use case diagrams and SPL FMs. Similarly, Griss *et al.* [35] integrated FMs with the reuse-driven Software Engineering process using a model-driven approach. They also aimed to construct a feature model based on requirements elicited from domain experts. While we use similar concepts in our approach, we also include derivation of the product, provide a mission configurator in addition to support for runtime configuration. Hajri *et al.* proposed PUMConf [36, 37], which provides tool-support for engineers configuring products from product line models. It guides the analyst through a series of configuration decision and automatically generates a use case and domain models based on the specified product. In contrast, we start from use case descriptions created in conjunction with stakeholders, transform them into mission-specific FMs and activity diagrams, which we then merge into PL-level models. Finally, Czarnecki *et al.* [19] maps features to a design model annotated with logic expressions associated with the features.

Product Derivation and Configurators: Finally, many researchers have presented solutions for deriving products from a defined PL. Deelstra *et al.* [20] presented a product derivation framework inspired by problems identified from an industrial case study. Rabiser and Dhungana [54] developed DOPLER using a decision-oriented variability modelling approach, which also used a series of predefined questions to configure a product.

8 CONCLUSIONS

The work described in this paper represents a pragmatic approach to developing a configurator that is well-suited for use by emergency responders on the field. We have described the process that we followed to collect and assimilate diverse use case scenarios into PL assets that were then used to develop and test our configurator. We have performed an initial user evaluation, and will take the lessons learned to improve the configurator and conduct field tests with firefighters and physical UAVs.

ACKNOWLEDGMENTS

The work was partially funded by the US National Science Foundation Grant CPS:1931962 and CCF:1513730.

REFERENCES

[1] 2018. Council Bluffs firefighters use drones in ice water rescue training,KETC Newswatch 7, https://www.youtube.com/watch?v=p2MdbrTgmso. (2018).

[2] A. Agrawal, S. Abraham, B. Burger, C. Christine, L. Fraser, J. Hoeksema, S. Hwang, E. Travnik, S. Kumar, W. Scheirer, J. Cleland-Huang, M. Vierhauser, R. Bauer, and S. Cox. 2020. The Next Generation of Human-Drone Partnerships: Co-Designing an Emergency Response System. In *Proc. of the 2020 Conf. on Human Factors in Computing Systems*.

[3] Oscar Alvear, Carlos T Calafate, Enrique Hernández, Juan-Carlos Cano, and Pietro Manzoni. 2015. Mobile pollution data sensing using UAVs. In *Proc. of the 13th Int'l Conf. on Advances in Mobile Computing and Multimedia*. 393–397.

[4] Luciano Baresi. 2018. Activity Diagrams. In *Encyclopedia of Database Systems, Second Edition*. https://doi.org/10.1007/978-1-4614-8265-9_9

[5] Don Batory, David Benavides, and Antonio Ruiz-Cortes. 2006. Automated analysis of feature models: challenges ahead. *Commun. ACM* 49, 12 (2006), 45–47.

[6] Abdurrahman Beg, Abdul Rahman Qureshi, Tarek Sheltami, and Ansar Yasar. 2020. UAV-enabled intelligent traffic policing and emergency response handling system for the smart city. *Personal and Ubiquitous Computing* (2020), 1–18.

[7] Antonia Bertolino, Alessandro Fantechi, Stefania Gnesi, and Giuseppe Lami. 2006. *Product Line Use Cases: Scenario-Based Specification and Testing of Requirements*. Springer Berlin Heidelberg, Berlin, Heidelberg, 425–445. https://doi.org/10.1007/978-3-540-33253-4_11

[8] Alexandre Braganca and Ricardo J Machado. 2007. Automating mappings between use case diagrams and feature models for software product lines. In *Proc. of the 11th Int'l Software Product Line Conf.* IEEE, 3–12.

[9] Stan Bühne, Günter Halmans, Kim Lauenroth, and Klaus Pohl. 2006. Scenario-based application requirements engineering. In *Software Product Lines*. Springer, 161–194.

[10] Gabriella Caroti, Andrea Piemonte, and Yari Pieracci. 2017. UAV-borne photogrammetric survey as USAR firefighter teams support. In *Proc. of the Int'l Conf. on Computational Science and Its Applications*. Springer, 3–15.

[11] Jessica R. Cauchard, Jane L. E, Kevin Y. Zhai, and James A. Landay. 2015. Drone & Me: An Exploration into Natural Human-Drone Interaction. In *Proc. of the 2015 ACM Int'l Joint Conf. on Pervasive and Ubiquitous Computing*. ACM, New York, 361–365.

[12] Chih-Chung Chang, Jia-Lin Wang, Chih-Yuan Chang, M. Liang, and Ming-Ren Lin. 2015. Development of a multicopter-carried whole air sampling apparatus and its applications in environmental studies. *Chemosphere* 144 (09 2015), 484–492.

[13] Andreas Claesson, Anders Bäckman, Mattias Ringh, Leif Svensson, Per Nordberg, Therese Djärv, and Jacob Hollenberg. 2017. Time to Delivery of an Automated External Defibrillator Using a Drone for Simulated Out-of-Hospital Cardiac Arrests vs Emergency Medical Services. *JAMA* 317, 22 (06 2017), 2332–2334.

[14] Jane Cleland-Huang and Ankit Agrawal. 2020. Human-Drone Interactions with Semi-Autonomous Cohorts of Collaborating Drones. In *Interdisciplinary Workshop on Human-Drone Interaction (iHDI 2020), CHI '20 Extended Abstracts, 26 April 2020, Honolulu, HI, US.*

[15] Jane Cleland-Huang and Michael Vierhauser. 2018. Discovering, Analyzing, and Managing Safety Stories in Agile Projects. In *Proc. of the 26th IEEE Int'l Requirements Engineering Conf.* 262–273. https://doi.org/10.1109/RE.2018.00034

[16] Jane Cleland-Huang, Michael Vierhauser, and Sean Bayley. 2018. Dronology: an Incubator for Cyber-Physical Systems Research. In *Proc. of the 40th Int'l Cong. on Software Engineering: New Ideas and Emerging Results*. 109–112. https://doi.org/10.1145/3183399.3183408

[17] Paul C. Clements and Linda Northrop. 2001. *Software Product Lines: Practices and Patterns*. Addison-Wesley.

[18] Alistair Cockburn. 2000. *Writing Effective Use Cases* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[19] Krzysztof Czarnecki and Michał Antkiewicz. 2005. Mapping features to models: A template approach based on superimposed variants. In *Proc. of the Int'l Conf. on Generative Programming and Component Engineering*. Springer, 422–437.

[20] Sybren Deelstra, Marco Sinnema, and Jan Bosch. 2005. Product derivation in software product families: a case study. *Journal of Systems and Software* 74, 2 (2005), 173 – 194. https://doi.org/10.1016/j.jss.2003.11.012

[21] Patrick Doherty, Fredrik Heintz, and David Landén. 2010. A distributed task specification language for mixed-initiative delegation. In *Proc. of the Int'l Conf. on Principles and Practice of Multi-Agent Systems*. Springer, 42–57.

[22] Mica R. Endsley. 2011. *Designing for Situation Awareness: An Approach to User-Centered Design, Second Edition* (2nd ed.). CRC Press, Inc., Boca Raton, FL, USA.

[23] K Anders Ericsson and Herbert A Simon. 1980. Verbal reports as data. *Psychological Review* 87, 3 (1980), 215.

[24] Magnus Eriksson, Jürgen Börstler, and Kjell Borg. 2005. The PLUSS approach–domain modeling with features, use cases and use case realizations. In *Proc. of the 9th Int'l Software Product Line Conf.* Springer, 33–44.

[25] Alessandro Fantechi and Stefania Gnesi. 2007. A behavioural model for product families. In *ESEC-FSE '07: Proc. of the the 6th Joint Meeting of the European Software Engineering Conf. and the ACM SIGSOFT Symp. on the Foundations of Software Engineering*. ACM, New York, NY, USA, 521–524.

[26] Alessandro Fantechi, Stefania Gnesi, Isabel John, Giuseppe Lami, and Jörg Dörr. 2004. Elicitation of Use Cases for Product Lines. In *Software Product-Family Engineering*, Frank J. van der Linden (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 152–167.

[27] Stefan Fischer, Lukas Linsbauer, Roberto Erick Lopez-Herrejon, and Alexander Egyed. 2014. Enhancing Clone-and-Own with Systematic Reuse for Developing Software Variants. *Proc. of the 2014 IEEE Int'l Conf. on Software Maintenance and Evolution* (2014), 391–400.

[28] Mathias Fleck. 2016. Usability of lightweight defibrillators for uav delivery. In *Proc. of the 2016 CHI Conf. Extended Abstracts on Human Factors in Computing Systems*. 3056–3061.

[29] Markus Funk. 2018. Human-drone interaction: Let's get ready for flying user interfaces! *Interactions* 25, 3 (2018), 78–81. https://doi.org/10.1145/3194317

[30] Sergio García, Patrizio Pelliccione, Claudio Menghi, Thorsten Berger, and Tomas Bures. 2019. High-level mission specification for multiple robots. In *Proc. of the 12th ACM SIGPLAN Int'l Conf on Software Language Engineering*. 127–140.

[31] Eddy Ghabach, Mireille Blay-Fornarino, Franjieh El Khoury, and Badih Baz. 2018. Guiding Clone-and-Own When Creating Unplanned Products from a Software Product Line. In *New Opportunities for Software Reuse*, Rafael Capilla, Barbara Gallina, and Carlos Cetina (Eds.). Springer International Publishing, Cham, 139–147.

[32] Khaled A Ghamry, Mohamed A Kamel, and Youmin Zhang. 2017. Multiple UAVs in forest fire fighting mission using particle swarm optimization. In *Proc. of the 2017 Int'l Conf. on Unmanned Aircraft Systems (ICUAS)*. IEEE, 1404–1409.

[33] Hassan Gomaa. 2004. *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[34] John C. Griffith and Ronald T. Wakeham. 2015. Unmanned Aerial Systems in the Fire Service: Concepts and Issues.

[35] Martin L Griss, John Favaro, and Massimo d'Alessandro. 1998. Integrating feature modeling with the RSEB. In *Proc. of the 5th Int'l Conf. on Software Reuse*. IEEE, 76–85.

[36] Ines Hajri, Arda Goknil, Lionel C Briand, and Thierry Stephany. 2016. PUMConf: a tool to configure product specific use case and domain models in a product line. In *Proc. of the 24th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering*. 1008–1012.

[37] Ines Hajri, Arda Goknil, Lionel C Briand, and Thierry Stephany. 2018. Configuring use case models in product families. *Software & Systems Modeling* 17, 3 (2018), 939–971.

[38] Günter Halmans and Klaus Pohl. 2003. Communicating the variability of a software-product family to customers. *Software and Systems Modeling* 2, 1 (2003), 15–36.

[39] Ivar Jacobson, Martin L. Griss, and Patrik Jonsson. 1997. *Software reuse - architecture, process and organization for business*. Addison-Wesley-Longman. I–XXVIII, 1–497 pages.

[40] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. 1990. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Technical Report. Carnegie-Mellon University Software Engineering Institute.

[41] Kyo C. Kang, Jaejoon Lee, and Patrick Donohoe. 2002. Feature-Oriented Product Line Engineering. *IEEE Software* 19, 4 (2002), 58–65. https://doi.org/10.1109/MS.2002.1020288

[42] Hyunbum Kim, Lynda Mokdad, and Jalel Ben-Othman. 2018. Designing UAV Surveillance Frameworks for Smart City and Extensive Ocean with Differential Perspectives. *IEEE Communications Magazine* 56, 4 (2018), 98–104. https://doi.org/10.1109/MCOM.2018.1700444

[43] Cengiz Koparan, Ali Bulent Koc, Charles V Privette, Calvin B Sawyer, and Julia L Sharp. 2018. Evaluation of a UAV-assisted autonomous water sampling. *Water* 10, 5 (2018), 655.

[44] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. 2009. Temporal-Logic-Based Reactive Mission and Motion Planning. *IEEE Transactions on Robotics* 25, 6 (2009), 1370–1381.

[45] Jacob Krüger, Sebastian Nielebock, Sebastian Krieter, Christian Diedrich, Thomas Leich, Gunter Saake, Sebastian Zug, and Frank Ortmeier. 2017. Beyond Software Product Lines: Variability Modeling in Cyber-Physical Systems. In *Proc. of the 21st Int'l Systems and Software Product Line Conf.* 237–241.

[46] H.T. Lally, I. O'Connor, O.P. Jensen, and C.T. Graham. 2019. Can drones be used to conduct water sampling in aquatic environments? A review. *Science of The Total Environment* 670 (2019), 569 – 575. https://doi.org/10.1016/j.scitotenv.2019.03.252

[47] Tomaz Mesar, Aaron Lessig, and David R King. 2019. Use of Drone Technology for Delivery of Medical Supplies During Prolonged Field Care. *Journal of special operations medicine : a peer reviewed journal for SOF medical professionals* 18 4 (2019), 34–35.

[48] Andrea Molino, Daniele Brevi, Guido Gavilanes, Riccardo Scopigno, Anooq Sheikh, and Enea Bagalini. 2016. Using drones for automatic monitoring of vehicular accident. In *Proc. of the 2016 AEIT Int'l Annual Conf.*

[49] Jakob Nielsen and Thomas K. Landauer. 1993. A Mathematical Model of the Finding of Usability Problems. In *Proc. of the INTERCHI '93 Conf. on Human Factors in Computing Systems (INTERCHI '93)*. IOS Press, NLD, 206–213.

[50] John-Paul Ore, Sebastian Elbaum, Amy Burgin, and Carrick Detweiler. 2015. Autonomous Aerial Water Sampling. *Journal of Field Robotics* 32, 8 (2015), 1095–1113. https://doi.org/10.1002/rob.21591 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21591

[51] Klaus Pohl, Günter Böckle, and Frank van der Linden. 2005. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[52] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. 2005. Jadex: A BDI reasoning engine. In *Multi-agent programming*. Springer, 149–174.

[53] Luís Pádua, José Sousa, Jakub Vanko, Jonáš Hruška, Telmo Adão, Emanuel Peres, António Sousa, and Joaquim J. Sousa. 2020. Digital Reconstitution of Road Traffic Accidents: A Flexible Methodology Relying on UAV Surveying and Complementary Strategies to Support Multiple Scenarios. *International Journal of Environmental Research and Public Health* 17, 6 (Mar 2020), 1868. https://doi.org/10.3390/ijerph17061868

[54] Rick Rabiser, Paul Grunbacher, and Deepak Dhungana. 2007. Supporting product derivation by adapting and augmenting variability models. In *Proc. of the 11th Int'l Software Product Line Conf.* IEEE, 141–150.

[55] Anand S Rao, Michael P Georgeff, et al. 1995. BDI agents: from theory to practice.. In *ICMAS*, Vol. 95. 312–319.

[56] Jennifer Rios. 2019. Firefighters practice using drones to assist ice rescues , URL: https://www.broomfieldenterprise.com/2019/01/11/firefighters-practice-using-drones-to-assist-ice-rescues/,Last referenced (2020-05-14). (2019).

[57] Jose Ruiz-Jimenez, Nicola Zanca, Hangzhen Lan, Matti Jussila, Kari Hartonen, and Marja-Liisa Riekkola. 2019. Aerial drone as a carrier for miniaturized air sampling systems. *Journal of Chromatography A* 1597 (04 2019). https://doi.org/10.1016/j.chroma.2019.04.009

[58] Philipp Schillinger, Mathias Bürger, and Dimos V. Dimarogonas. 2018. Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. *The International Journal of Robotics Research* 37, 7 (2018), 818–838. https://doi.org/10.1177/0278364918774135 arXiv:https://doi.org/10.1177/0278364918774135

[59] SEI, Software Engineering Institute. 2020. Software Product Lines. http://www.sei.cmu.edu/productlines. (2020).

[60] Vítor E Silva Souza, Alexei Lapouchnian, William N Robinson, and John Mylopoulos. 2011. Awareness requirements for adaptive systems. In *Proc. of the 6th Int'l Symposium on Software Engineering for Adaptive and Self-managing Systems*. 60–69.

[61] Mario Silvagni, Andrea Tonoli, Enrico Zenerino, and Marcello Chiaberge. 2017. Multipurpose UAV for search and rescue operations in mountain avalanche events. *Geomatics, Natural Hazards and Risk* 8, 1 (2017), 18–33. https://doi.org/10.1080/19475705.2016.1238852 arXiv:https://doi.org/10.1080/19475705.2016.1238852

[62] David R Thomas. 2006. A general inductive approach for analyzing qualitative evaluation data. *American journal of evaluation* 27, 2 (2006), 237–246.

[63] Ulrike Thomas, Gerd Hirzinger, Bernhard Rumpe, Christoph Schulze, and Andreas Wortmann. 2013. A new skill based robot programming language using uml/p statecharts. In *Proc. of the 2013 IEEE Int'l Conf. on Robotics and Automation*. IEEE, 461–466.

[64] Michael Vierhauser, Jane Cleland-Huang, Sean Bayley, Thomas Krismayer, Rick Rabiser, and Paul Grünbacher. 2018. Monitoring CPS at Runtime - A Case Study in the UAV Domain. In *Proc. of the 44th Euromicro Conf. on Software Engineering and Advanced Applications, SEAA 2018, Prague, Czech Republic, August 29-31, 2018*. 73–80. https://doi.org/10.1109/SEAA.2018.00022

[65] D.M. Weiss and C.T.R. Lai. 1999. *Software product-line engineering: a family-based software development process.* Addison-Wesley.

[66] Jon Whittle, Pete Sawyer, Nelly Bencomo, Betty HC Cheng, and Jean-Michel Bruel. 2010. RELAX: a language to address uncertainty in self-adaptive systems requirement. *Requirements Engineering* 15, 2 (2010), 177–196.

[67] Michael Wooldridge. 1997. Agent-based software engineering. *IEE Proceedings-software* 144, 1 (1997), 26–37.

[68] Nan Zhao, Weidang Lu, Min Sheng, Yunfei Chen, Jie Tang, F Richard Yu, and Kai-Kit Wong. 2019. UAV-assisted emergency networks in disasters. *IEEE Wireless Communications* 26, 1 (2019), 45–51.

[69] SD Zhi, YB Wei, and ZH Yu. 2017. Air quality monitoring platform based on remote unmanned aerial vehicle with wireless communication. In *Proc. of the Int'l Conf. on Future Networks and Distributed Systems*. 1–7.