1. What is Dronology about?

**Dronology**is an Open-Source Unmanned Aerial System (**UAS**) management and control system that **supports**the flights of **multiple**physical or simulated drones.

The **Dronology**project is based at the University of Notre Dame's Computer Science and Engineering Department. It was launched in January 2017 and is currently in the incubation stage.

**Dronology**is managed in **GitHub**(currently in a private repository), with the goal of full public release as the project proceeds. Further, the project will release research bundles — containing artifacts packaged in ways that support experimentation across diverse areas of Software Engineering.

The project has **two**main objectives:

1. **First**, to **establish a research environment** for studying various software and systems engineering aspects for cyber-physical systems.
2. **Second**, to **provide a framework for controlling and coordinating the flight** of individual **UAS**, formations, and swarms to support applications such as search-and-rescue, surveillance, and scientific data collection.

**Unmanned Aerial Systems**(**UAS**) are becoming increasingly prevalent in today's world. They are used to film sporting events, deliver commercial packages, provide medical supplies, survey farmland, and collect air samples.

Many commercial systems have come on the market for managing **UAS**flights; however, these systems **tend to focus**on planning, monitoring, and controlling the flights of a **single UAS.** In contrast, **Dronology**focuses on coordinating UAS in tasks where they either work **collaboratively**to achieve a goal or perform independent functions in the same airspace.

The **Dronology architecture**is designed to deliver a multi-platform solution. It is designed around the concept of a Dronology Core with Services middleware that supports UI development and Ground station middleware that supports communication with diverse **UAS**.

Drone applications are **Safety-Critical** because they are software systems whose failure or malfunction will likely result in death or serious injury, loss or severe damage to equipment or property, or environmental harm.

**Dronology**is funded under the National Science Foundation.

**Dronology**is first and foremost a research environment for investigating challenges in safety-critical software systems.

The **Dronology**project has several vital interests:

**Safety**

* This is challenging in shared airspace with the potential for multiple, independently operated UAS.

**Runtime Monitoring**

* Where sUAS are monitored at runtime to ensure safety compliance.

**Onboard Intelligence**

* Where sUAS are equipped to make intelligent runtime decisions that enable them to adapt to current contexts.

**Core components for Dronology:**

**Dronology Core**

* **Dronology Core**provides features for coordinating the flights of multiple sUAS in shared airspace. This includes centralized collision avoidance, fleet management, internal simulator, and vehicle status management. Click here to learn more about critical components in the Dronology core.

**Dronology services**

* Various clients use **Dronology services**to construct user interfaces. The services can perform tasks such as uploading flight plans, activating UASs, observing and visualizing UAS flights, and registering equipment to individual UASs.

**Vaadin UI**

**Vaadin**is an open-source web framework that supports the development of rich user interfaces. Its server-side architecture allows most of the logic to run on the server, while Ajax technology on the client side ensures a rich and highly interactive UI. In **Dronology**, we run **Vaadin**on **TomCat**— which can be deployed either centrally or locally. The web-based approach enables our goal of platform portability.

**Ground station middleware**

* The ground station middleware bridges the Dronology core and a diverse set of ground stations. It is designed to allow future developers to build their ground stations. The middleware enables the registration of a new ground station. It passes directives from the **Dronology core** to **UAVs**registered through their respective ground stations and forwards UAV data, such as their **GPS coordinates**, from the ground station back to the core.

**Ground station**

* **Ground stations** are responsible for communicating directly with the physically or externally simulated UAVs. Our current ground station is developed in **Python**using **Dronekit Python**— and can communicate with UAVs that run **ArduPilot**.

**Runtime monitoring**

The **runtime monitoring** component allows constraints to be defined and various values to be monitored. The monitored variables include UAV-specific values such as coordinates, velocity, and battery power and external variables read from sensors on the UAV.

b. What are its main features of it?

**Dronology**is currently **under development** in an **incubation**phase. It already has many features that work, but some are under development. The **official website** shows a few of the main features:

**Core Dronology Controller**

* Dispatches and tracks UAS flights.
* Manages physical and simulated **UAS**interchangeably, enabling scalable or mixed physical/simulated experiments.
* Supports both waypoints and vector commands.
* Written in Java.

**Ground Control Station**

* Serves as a **communication bridge**between the **Dronology**controller and **UAVs**.
* Detects physical UAVs that are pre-registered with the system.
* Receives status messages from **UAVs**(e.g., current location, battery level, velocity) and sends commands from **Dronology**to the UAVs.
* Written in **Python**and communicates to UAVs using DroneKit Python (based on MavLink).
* It communicates to **Dronology**via middleware and is, therefore, easily replaceable.

**User Interface**

* Middleware for UI development - allows us to register for events and to create and submit routing information to **Dronology**.

**Safety**

* Well, we aren't there yet. **Collision**avoidance is currently **under construction**.

There are also **planned features** as:

* Collision avoidance (currently in progress)
* Swarming
* Greater flexibility - to allow various features to be replaced (statically and/or at runtime)

c. What are the stakeholders of Dronology? (Please consider the Stakeholder Onion Diagram)

A **stakeholder**is an "individual or group that has an interest in the activity of an organization, strategy, initiative or project. A stakeholder can either **affect**or **be affected** by the business. Stakeholders can be internal or external.

The stakeholders of Dronology are:

* Team Leader: Jane Cleland-Huang
* Chief Architect: Dr. Michael Vierhauser
* Hardware arm leader: Dr. Jane Wyngaard
* UI developer: Dr. Jinghui Cheng
* Software developer: Michael Murphy
* Researcher: Sean Bayley
* Undergraduate students at Notre Dame or elsewhere in the USA, or post-docs.
* Collaborative researchers: Greg Madey, Vijay Gupta, Christian Poellabauer, Dong Wang
* Advisory board: Martin Lehofer (Head of Systems Integration Research Group at Siemens Corporate Technology)
* External collaborators: Christian Doppler Laboratory on Monitoring and Evolution of Very-Large-Scale Software Systems, Software Engineering and Design at Open University
* Emergency responders
* FAA regulators/FAA
* General public/public
* RPIC
* Victims/people who need help
* Fire department
* Environmentalists
* Farmers
* Mission Operator
* Technicians
* Mission Commander(human)
* Air Traffic Control
* Human Operator
* Drone Commander
* Clients

d. What are the available artefacts, and what are they used for?

The available **artifacts**are:

**Use case - A use case describes how a user uses a system to accomplish a particular goal.**

* Found in the Excel document provided.
* **What it is used for:** Use cases help brainstorm what could go wrong. It may establish the success and failure scenarios and any critical variations or exceptions. Use cases to help manage scope, establish requirements, communicate technical requirements, and outline how users will interact with the system or risk management.
* **In our case:**There are **5**use cases and **11**supporting use cases, each with the main success scenario and describing in detail and technical terms what the software should do. Some steps from use cases are linked to supporting use cases for further explanations.

**Software documentation - This information describes the product to those who develop, deploy, and use it. It includes technical manuals and online material, such as online versions of manuals and helps capabilities.**

* **Found in** materials provided like Excel document and ICSE-NIER paper.
* **What it is used for:**The presence of documentation helps keep track of all aspects of an application and improves the software product's quality. Helps development teams during development—assists in establishing internal coordination in work. Explain product functionality and allow for discussing all significant questions arising between stakeholders and developers. It gets users familiar with the software and makes them aware of its feature.
* **In our case:**We have the excel document, ICSE-NIER paper, course slides, seminar slides, and Dronology's website.

**Requirements - Any necessary (or sometimes desired) function, attribute, capability, characteristic, or quality of a system to have value and utility to a customer, organization, internal user, or other stakeholders.**

* **Found**in the Excel document provided.
* **What it is used for:** Requirements management aims to ensure product development goals are successfully met. They help at getting a deeper understanding of the future product. Clearly define deliverables and build only relevant functionality. Requirements specify a product's features and how those features should work.
* **In our case:** We have a "Requirements" file in the Excel document that describes the goal and what should happen when specific things are triggered.

**Notes – Small text boxes.**

* **Found**in descriptions of use cases in the Excel document provided.
* **It is used for:**Giving further explanations or clarifying what something means.
* **In our case:** For every use case, there is a small text box explaining the meaning of "Main Success Scenario".

**Failure Assessments – Analysis of potential failure(s) which may happen.**

* **Found**in descriptions of use cases in the Excel document provided.
* **What it is used for:**Enable developers to anticipate the poorly working system and take this situation into account when developing the product. In that way, developers can think about possible solutions to these problems by rethinking the product or fixing them after they happen by having designed steps for troubleshooting.
* **In our case:**We have a "Failure end condition" described in every use case and supporting use case.

**ICSE-NIER paper – Scientific paper.**

* **Found**in the materials provided.
* **What it is used for:** Exposing the problem that most existing approaches lack support for the automated generation and setup of monitors and constraints for diverse technologies and do not provide adequate support for evolving the monitoring infrastructure. Furthermore, the authors of this paper come with a new piece of the software described.

**Class diagram - This type of static structure diagram describes a system's structure by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. It is a way to map out the structure of a piece of software or application.**

* **Found**in the ICSE-NIER paper provided in chapters II and III.
* **What it is used for:** Illustrate data models for information systems, no matter how simple or complex. Class diagrams give a better understanding of the general overview of the schematics of an application. Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure. It describes the primary responsibilities of a system. It provides an overview of how an application is structured before coding. It allows the individuals to conclude the kind of implantation procedures required in the system for better outcomes. It helps with a more accessible and clear understanding of the overview and the schematics involved with the overall system or the process.
* **In our case:** We have a diagram that overviews ModIRMo's main component. It is shown in chapter III of the scientific paper.

**Components document - Excel document for components of the system.**

* **Found**in the Excel document provided in the "Components" file.
* **What it is used for:** Highlighting core components of the system and describing the responsibility for each of them.
* **In our case:** We have a "Component" file in the Excel provided describing each component's role.

**Source Code – The logic behind a piece of software.**

* **Found**in the ICSE-NIER paper provided in chapter IV.
* **What it is used for:**Used in explanations in ICSE-NIER paper.
* **In our case:** We have a code snippet in chapter IV, an example of a VIATRA constraint for checking the number of satellites.

**Prototypes – A model/sample/release created to test the concept.**

* **Found**in course slides where we were presented with a video about UAS and on the Dronology website in the "Projects" section. There are two projects: River Search and Rescue and Defibrillator delivery.
* **What it is used for:** They help a developer see what has been done and give them an idea of where to go next.
* **In our case:** We have the course slides where we were presented with a sample and Dronology's website.

**Unified Modeling Language (UML) - UML is a way of visualizing and plotting how a piece of software works. It works to map out links and processes. Serves as a Visual Representation between Classes and Entities.**

* **Found**in the ICSE-NIER paper provided in chapter III.
* **What it is used for:** UML allows different software developers to work on the same project by providing a common language. This enhances collaboration and provides for a more efficient design process. It also helps identify potential problems early in the design process. UML allows developers to see the big picture of a system. This can help them understand the system better and identify potential challenges. It also allows them to see how different system parts interact. UML provides a standard way to design software and systems. This allows for a more unified design process and helps to ensure compliance with standards. It also offers more accessible design reviews and enhanced productivity.
* **In our case:** We have a UML example in chapter III which shows the Monitoring Meta-Model used in ModIRMo for specifying parts of the monitored system.

**Images – Photos with the product.**

* **Found**on the official website of Dronology.
* **What it is used for:** Images are used to help design the product or preliminary design images. They could even be simple sketches and diagrams used to help map out the software.
* **In our case:** We have images on Dronology's website.

**Constraint - Rigors that must be respected.**

* Found in the ICSE-NIER paper provided in chapter V.
* **What it is used for:**It is used to ensure a safer take-off and minimize risks.
* **In our case:** In chapter V of the scientific paper, the authors showed us UAV constraints used with Dronology simulations.