Dronology: An Incubator for Cyber-Physical Systems Research

Jane Cleland-Huang, Michael Vierhauser, Sean Bayley
Department of Computer Science and Engineering
University of Notre Dame
South Bend, IN, USA
{JaneClelandHuang,mvierhau,sbayley}@nd.edu

ABSTRACT

Research in the area of Cyber-Physical Systems (CPS) is hampered by the lack of available project environments in which to explore open challenges and to propose and rigorously evaluate solutions. In this "New Ideas and Emerging Results" paper we introduce a CPS research incubator – based upon a system, and its associated project environment, for managing and coordinating the flight of small Unmanned Aerial Systems (sUAS). The research incubator provides a new community resource, making available diverse, high-quality project artifacts produced across multiple releases of a safety-critical CPS. It enables researchers to experiment with their own novel solutions within a fully-executable runtime environment that supports both high-fidelity sUAS simulations as well as physical sUAS. Early collaborators from the software engineering community have shown broad and enthusiastic support for the project and its role as a research incubator, and have indicated their intention to leverage the environment to address their own research areas of goal modeling, runtime adaptation, safety-assurance, and software evolution.

CCS CONCEPTS

• **Software and its engineering → Software safety**;

KEYWORDS

Research Environments, Unmanned Autonomous Systems, Software Engineering

ACM Reference Format:
Jane Cleland-Huang, Michael Vierhauser, Sean Bayley. 2018. Dronology: An Incubator for Cyber-Physical Systems Research. In *ICSE-NIER'18: 40th International Conference on Software Engineering: New Ideas and Emerging Results Track, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3183399.3183408

1 INTRODUCTION

The emerging adoption of Cyber-Physical Systems (CPS) in areas ranging across autonomous driving, smart cities, and unmanned aerial systems makes it imperative to address software engineering (SE) challenges for safety-critical systems [3, 20]. Challenges are diverse and include goal modeling [4], formal methods, requirements completeness [13], runtime adaptation, [2], design-time evolution [12], product line development [16], and human-computer interaction, to name a few. However, such research cannot be conducted in a vacuum, and must be motivated, explored, and evaluated within the context of realistic and real-world systems.

The Dronology system, and its associated project environment serves as a *research incubator* to facilitate and nurture early growth of research ideas in the CPS domain. It enables researchers to experiment with a theory or hypothesis, in a controlled environment, and to progressively develop the idea until it is ready for testing and deployment in a full industrial setting. Currently many software engineering concepts do not progress past inception because researchers lack the means to advance them through intermediate stages of growth. Dronology provides a full *project environment* (i.e., a fully functional open-source system) for managing, monitoring, and coordinating the flights of multiple small Unmanned Aerial Systems (sUAS) and a *dataset* including diverse, high-quality project artifacts produced across multiple releases. As such, it provides a realistic incubator for investigating CPS topics, and for enabling researchers to experiment with their own innovative solutions within a robust project environment. Furthermore, as sUAS accidents, such as in-air collisions, or crash landings into vehicles or pedestrians, have the potential for causing significant harm to people and/or property damage. Dronology is therefore considered *safety critical.*

To illustrate the need for a CPS research incubator, consider a researcher working in the space of trace link evolution and reuse in order to support safety analysis in a safety-critical product line. His work is greatly impeded by the lack of publicly available, non-trivially sized, multi-version, software systems containing the type of diverse software artifacts and feature definitions that are expected in a safety-critical product line. Similarly, a researcher working in the space of CPS runtime adaptation has developed her own experimental environment for proof-of-concept exploration; however, she is interested in evaluating her work on runtime composition of features in a larger-scale, real-world environment. The current lack of such a publicly available environment is a significant impediment to research in the CPS space. We aim to address this issue through our proposed CPS research incubator.

In the remainder of this paper, Section 2 discusses shortcomings of existing community resources. Sections 3 and 4, describe the Dronology project environment and provide examples of research areas that the incubator is designed to support. Finally, in Section 5 we describe ongoing goals for our proposed research incubator.

2 EXISTING RESEARCH ENVIRONMENTS

One of the most closely related research contributions is the PROMISE repository established by Menzies et al. [18]. PROMISE includes an

ICSE-NIER'18, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5662-6/18/05…$15.00
https://doi.org/10.1145/3183399.3183408

extensive collection of diverse software engineering artifacts – contributed by numerous SE researchers and used for countless experiments. However, a major limitation of the repository with respect to CPS research stems from the fact that the hosted datasets tend to represent simple snapshots of specific parts of a system and neither provide executable code nor diverse project artifacts. Recent history has taught us that researchers actively pursue open challenges that are supported by appropriate datasets. A compelling example is the concurrent growth of open-source system (OSS) projects with increased research emphasis on *mining software repositories* (MSR). Our goal is to empower CPS researchers in the same way that OSS empowered MSR researchers. In the remainder of this section we summarize the reasons that current datasets and environments are insufficient to meet CPS research needs.

Open Source Systems: OSS projects provide a rich source of data for many areas of investigation. However they are more suited to research that focuses primarily on source code and related artifacts, such as bug reports, test cases, or developer interactions, and do not provide suitable environments for studying other types of challenges, such as the tight interaction between hardware and software that is prevalent in safety-critical and/or Cyber-Physical Systems.

Existing Community Datasets: Several community datasets support focused research efforts. For example the Generic Infusion Pump (GIP), includes requirements, architectural designs, and formal models and has been used in many studies [19]. However, its artifacts form a relatively disjoint set and do not include source code. The iTrust system includes use cases, source code, and limited (or outdated) trace links [24] but represents an IT system and is therefore also not suited to CPS research. Similarly, datasets provided by PROMISE [18], the Center of Excellence for Software Traceability [6], and the SIRS testing repository [14], are widely used by software engineering researchers but only provide data snapshots and lack complete and comprehensive project environments.

Industry Datasets: Industrial datasets provide rich research environments. However, finding industry collaborators can be challenging, and data that is shared under non-disclosure agreements can not be shared openly, meaning that researchers can not compare their solutions or reveal meaningful contextual details. Therefore, a pressing need exists for a publicly accessible safety-critical CPS project that provides a rich project context for enabling experimentation across the diverse challenges of CPSs.

3 DRONOLOGY: A RESEARCH INCUBATOR

We established criteria for the CPS research incubator that it must (1) include a cyber-physical component, (2) use affordable and readily available hardware, (3) be accessible to a broad set of researchers in terms of the necessary domain knowledge, (4) include diverse and challenging safety hazards, and (5) be developed using standards commensurate with medium criticality projects.

3.1 Dronology Product

Dronology is developed as a Java-based framework for controlling, managing, monitoring, and coordinating flights of sUAS in an urban airspace. It is compatible with ArduPilot-based[1] sUAS and integrates a high-fidelity Software-in-the-loop (SITL) simulator as well as physical sUAS via a Python-based groundstation.

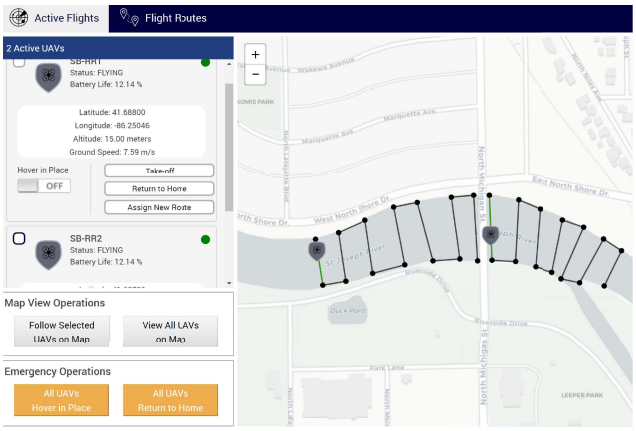

Figure 1: Dronology: sUAS flight routes displayed and monitored in real-time

To develop a realistic, industrial-strength CPS environment we are working with end-users on two concrete projects to support river rescue and defibrillator delivery. Figure 1 illustrates the river-rescue scenario in our Dronology UI. We are also working closely with researchers from four diverse research groups to ensure that the Dronology architecture, process, and artifacts actually support researcher's needs in their areas of interest.

The project produces diverse artifacts such as hazards, faults, environmental assumptions, requirements, design definitions, architectural design, a feature model, use cases, acceptance tests with test logs, unit tests, source code, and bug reports as shown in Figure 2. Though incomplete, this set of artifacts is extensible and represents fundamental elements of safety-critical systems [21]. Version 0.1, which will be released in Spring of 2018, currently includes approximately 20,000 lines of code (in around 300 Java classes and 30 Python classes) a feature model containing 39 features, around 250 requirements alongside with 150 design definitions, 95 bug reports, numerous safety-related artifacts, and the architecture shown in Figure 3. A current description of the project goals, its current status, sample artifacts, process description, and release timeline is available at http://www.dronology.info.

The project environment will be available upon request via github, where researchers can download or fork the project to build their own new features, extensions, and modifications. In the future it will be fully open-sourced. As it is common in many open-source projects we will establish a review process to integrate contributions of newly developed features that warrant inclusion in future official releases.

3.2 Dronology Process

Dronology seeks to deliver a high-quality, industrial-like project environment. To this end, our initial project team includes a hired professional developer with previous experience working on sUAS

[1] http://ardupilot.org

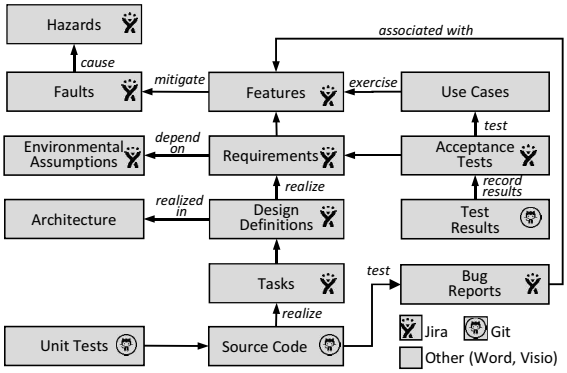

Figure 2: Dronology' Artifacts and Traceability Paths

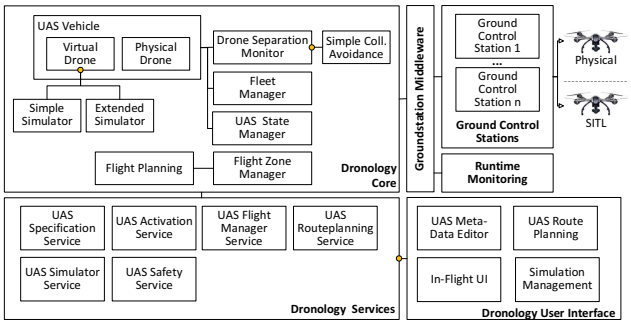

Figure 3: High level Architecture of Dronology

applications, four team members with significant industrial programming experience, a hardware expert who is a qualified sUAS pilot, one team member with extensive experience working in safety-critical systems, in addition to several graduate and undergraduate students who have completed internships on the project. Finally, we are establishing an advisory board of actively engaged industrial experts, some of whom have already provided feedback on the project.

The difference between our system and industrial systems is therefore not in the scope, completeness, or quality of the delivered system, but rather in the goals that drive its development. Our goals are two-fold. First, to *build a safe, deployed, working system* in conjunction with real end-users, and second, to *create and maintain a research incubator that meets the needs* of researchers working in diverse areas of safety-critical systems and cyber-physical systems.

3.3 Limitations

While we are convinced that our research incubator supports the needs of researchers in diverse areas, there are certain limitations that need to be taken into consideration. For example, using Java and Python as our main implementation languages might limit the suitability for certain areas of real-time critical CPS research, meaning that in its current form Dronology does not represent a system which exhibits hard real-time characteristics. While there have been efforts to use Java in real time embedded systems [22] this is out of scope of the (core) application area of the research incubator. However, the product-line approach allows us to add additional features or alternative implementations later as needed.

4 SUPPORTED RESEARCH AREAS

Research challenges associated with CPS and other types of safety-critical system are diverse. We present six topic areas and briefly describe aspects of the Dronology incubator that would support research in each area. This list is not intended to be complete.

• **Software and Systems Requirements:** In their roadmap paper, Nuseibeh and Easterbrook [20] outlined specific challenges associated with eliciting, specifying, analyzing, modeling, and using requirements. Almost all of these challenges are relevant in the context of CPS. Furthermore, several specific problems, such as requirements analysis and modeling, are particularly impacted by the CPS nature of the project and should be explored within such an environment. The Dronology environment creates opportunities for research in several of these areas, for example, creating requirements models that represent data, domain, and behavioral views of the system. Dronology includes goals, system and software level requirements, and design definitions that provide necessary foundations for requirements analysis and modeling.

• **Software Traceability:** Software traceability captures relationships between uniquely identifiable software engineering artifacts and is required for certification (e.g., DO-178C) in most safety-critical domains. Specific challenges laid out in a traceability roadmap on "trends and future directions" include trace strategizing, trace link creation and evolution, and trace link usage and visualization [7]. The diversity of artifacts in the Dronology project environment, coupled with its extensive network of trace links, and the project's full history of versions and commits enables broad research across all open research areas. Such extensive and diverse sets of artifacts are not available in any datasets currently provided by the Center of Excellence for Software Traceability [6] or in any other dataset we are aware of.

• **CPS Product Lines:** Many CPSs could appropriately be developed as product lines in which a set of products that share a common, managed set of features are developed from a common set of core assets in a prescribed way [8]. Researchers in this area address diverse topics at both the domain and application engineering levels [16] including variability modeling and analysis, consistency between applications and domain artifacts [23], formal verification and testing, as well as application realization, feature interactions [1], dynamic product generation, and safety assurance of individual products. Datasets provided within the Product Line (PL) community tend to focus on feature models and do not provide complete project environments. Dronology is developed as a CPS product line with a respective feature model, PL architecture, variability points, and a configuration mechanism, and therefore provides an environment for PL experimentation.

• **Safety Assurance:** Most CPSs are safety-critical – leading to research in areas of formal modeling and verification, testing, simulation, hazard analysis, and safety assurance [17]. In this space, systems can be developed both formally and informally – although both approaches require a rigorous, safety-imbued engineering

process. The Dronology process includes a preliminary hazard analysis, a failure model effect criticality analysis (FMECA), mitigating requirements that address functional concerns and architectural solutions such as fault-tolerance, performance, and reliability, and Safety Assurance Cases [15]. We plan a limited set of formal models (e.g., in the areas of collision avoidance and obstacle detection); however researchers could build their own models based on Dronology requirements, design definitions, and associated properties.

• **Runtime Monitoring and Adaption:** CPSs often need to adapt at runtime to changes in their environments. This introduces a complex and multidisciplinary research agenda [10], that includes modeling and monitoring [2] causes of adaptation, specifying and constructing mechanisms of adaptation, and analyzing effects of adaptation upon the system and its safety [5]. Dronology provides an extensible runtime monitoring infrastructure allowing researchers to monitor diverse data collected from UAV sensors as well as internally generated events such as flight mode transitions. Adaptation strategies can be evaluated at variability points in the product line or through modifying cloned source code.

• **Human Studies:** Last, but not least, is the opportunity for supporting Human Studies within the context of safety-critical CPS. Such studies are also limited by a lack of immersive environments. Example research topics include studying visualization techniques during change impact analysis [9], studying the role of humans in trace link creation and maintenance [11], and studying ways in which humans leverage tools to analyze system safety. The Dronology project environment creates a viable context for such studies.

5 FUTURE VISION

We have introduced our vision for Dronology as a research incubator for safety-critical CPS. Future activities include: (1) *Ongoing development* of numerous features to be delivered across multiple versions; (2) A *product line* with multiple variability points and assets to be delivered at both the domain and application level; (3) *Research resources* to facilitate research challenges identified by the community, by organizing artifacts as customized downloadable data bundles; (4) An *atypical OSS* for which contributors will provide not just source code, but other safety-related artifacts required by the Dronology project; (5) *real end users* to bring realism to the project; (6) *Incremental releases* starting in May, 2018.

Our research incubator goes far beyond a more traditional dataset consisting of snapshots of data. It provides a rich project environment that includes diverse artifacts commensurate with a safety-critical domain. Our incubator is designed to empower researchers working in areas of CPS research which suffer from a dearth of experimental project environments. Dronology is planned as a community endeavor and collaborators are welcomed to work with us to make this a long-term, effective community resource. Project artifacts may be downloaded directly from http://www.dronology.info while source code is available upon request by research groups.

ACKNOWLEDGMENTS

This project has been funded by the Austrian Science Fund (FWF) (J3998-N319) and US National Science Foundation Grants (CCF-1741781, CCF-1649448)

REFERENCES

[1] Sven Apel, Hendrik Speidel, Philipp Wendler, Alexander von Rhein, and Dirk Beyer. 2011. Detection of Feature Interactions Using Feature-aware Verification. In *Proc. of the 2011 26th IEEE/ACM Int'l Conf. on Automated Software Engineering*. IEEE, 372–375.

[2] Davi Monteiro Barbosa, Romulo Gadelha De Moura Lima, Paulo Henrique Mendes Maia, and Evilasio Costa. 2017. Lotus@Runtime: A Tool for Runtime Monitoring and Verification of Self-Adaptive Systems. In *Proc. of the 12th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems*. IEEE.

[3] Nelly Bencomo, Robert B. France, Betty H. C. Cheng, and Uwe Aßmann (Eds.). 2014. *Models@run.time - Foundations, Applications, and Roadmaps [Dagstuhl Seminar 11481, 2011]*. Lecture Notes in Computer Science, Vol. 8378. Springer.

[4] Antoine Cailliau and Axel van Lamsweerde. 2013. Assessing requirements-related risks through probabilistic goals and obstacles. *Requir. Eng.* 18, 2 (2013), 129–146.

[5] Radu Calinescu, Carlo Ghezzi, Marta Kwiatkowska, and Raffaela Mirandola. 2012. Self-adaptive software needs quantitative verification at runtime. *Commun. ACM* 55, 9 (2012), 69–77.

[6] Jane Cleland-Huang, Adam Czauderna, and Jane Huffman Hayes. 2013. Using tracelab to design, execute, and baseline empirical requirements engineering experiments. In *Proc. of the 21st IEEE Intn'l Requirements Eng. Conf.* 338–339.

[7] Jane Cleland-Huang, Orlena Gotel, Jane Huffman Hayes, Patrick Mäder, and Andrea Zisman. 2014. Software traceability: trends and future directions. In *Proc. of the on Future of Software Engineering*. 55–69.

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

[9] Jose Luis de la Vara, Markus Borg, Krzysztof Wnuk, and Leon Moonen. 2016. An Industrial Survey of Safety Evidence Change Impact Analysis Practice. *IEEE Trans. Software Eng.* 42, 12 (2016), 1095–1117.

[10] Rogério De Lemos, Holger Giese, Hausi A Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, et al. 2013. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*. Springer, 1–32.

[11] Alex Dekhtyar and Jane Huffman Hayes. 2012. Studying the Role of Humans in the Traceability Loop. In *Software and Systems Traceability*. 241–261.

[12] Thomas R. Devine, Katerina Goseva-Popstojanova, Sandeep Krishnan, and Robyn R. Lutz. 2016. Assessment and cross-product prediction of software product line quality: accounting for reuse across products, over multiple releases. *Autom. Softw. Eng.* 23, 2 (2016), 253–302.

[13] Byron DeVries and Betty H. C. Cheng. 2016. Automatic detection of incomplete requirements via symbolic analysis. In *Proc. of the ACM/IEEE 19th Int'l Conf. on Model Driven Engineering Languages and Systems*. 385–395.

[14] Hyunsook Do, Sebastian Elbaum, and Gregg Rothermel. 2005. Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and Its Potential Impact. *Empirical Softw. Eng.* 10, 4 (Oct. 2005), 405–435.

[15] Tim Kelly and Rob Weaver. 2004. The Goal Structuring Notation–a safety argument notation. In *WS on Assurance Cases of Dependable Systems and Networks*.

[16] 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.* ACM, 237–241.

[17] Nancy G. Leveson. 1995. *Safeware: System Safety and Computers*. ACM, New York, NY, USA.

[18] Tim Menzies, R. Krishna, and D. Pryor. 2016. The Promise Repository of Empirical Software Engineering Data". (2016).

[19] Anitha Murugesan, Michael W. Whalen, Sanjai Rayadurgam, and Mats Per Erik Heimdahl. 2013. Compositional verification of a medical device system. In *ACM SIGAda Annual Conf. on High integrity language technology*. 51–64.

[20] Bashar Nuseibeh and Steve M. Easterbrook. 2000. Requirements Engineering: a Roadmap. In *Proc. of the 22nd Int'l Conf. on on Software Engineering, Future of Software Engineering Track*. 35–46.

[21] Patrick Rempel, Patrick Mäder, Tobias Kuschke, and Jane Cleland-Huang. 2014. Mind the gap: assessing the conformance of software traceability to relevant guidelines. In *Proc. of the Int'l Conf. on Software Engineering*. 943–954.

[22] D. C. Sharp, E. Pla, K. R. Luecke, and R. J. Hassan. 2003. Evaluating real-time Java for mission-critical large-scale embedded systems. In *Proc. of the 9th IEEE Real-Time and Embedded Technology and Applications Symp.* 30–36.

[23] Michael Vierhauser, Paul Grünbacher, Alexander Egyed, Rick Rabiser, and Wolfgang Heider. 2010. Flexible and scalable consistency checking on product line variability models. In *Proc. of the IEEE/ACM Int'l Conf. on Automated software engineering*. ACM, 63–72.

[24] Laurie Williams and Yonghee Shin. 2006. Work in progress: Exploring security and privacy concepts through the development and testing of the iTrust medical records system. In *Proc. of the 36th Annual Conf. on Frontiers in Education*. IEEE, 30–31.