Exploring Requirement Specification for Software that Learns

Anastasia Mavridou[1], Marie Farrell[2], Johann Schumann[1], Tom Pressburger[3]

[1]KBR Inc. / NASA Ames Research Center, USA
[2]University of Manchester, UK
[3]NASA Ames Research Center, USA
anastasia.mavridou@nasa.gov

Abstract

The development of software that learns has revolutionized how many systems perform. For the most part, these systems are neither safety- nor mission-critical. However, as technology and aspirations advance, there is an increased desire and need for Machine Learning (ML) software in safety- and mission-critical systems, e.g., driverless cars or autonomous space robotics. In these domains, reliability is crucial and systems have to undergo much scrutiny in terms of both the developed artefacts and the adopted development process. Central to the development of such systems is the elicitation and definition of software requirements that are used to guide the design and verification process. The addition of software components that learn, and the associated capability for unforeseen behavior, makes defining detailed software requirements challenging. In this talk, we examine requirements for ML components and identify key characteristics. We also present ongoing work on extending NASA's Formal Requirement Elicitation Tool (FRET) for capturing and formalizing requirements for software that learns.

Introduction

The design of critical systems begins with the definition of natural-language objectives and high-level requirements. Once defined, high-level objectives and requirements are subsequently decomposed into detailed system- and module-level requirements. Any subsequent verification and validation effort must support traceability of requirements and provide evidence that they are upheld. In fact, requirements traceability is prescribed by a number of international standards including DO-178C for the aerospace domain (Rierson 2017).

This process is well-understood for traditional systems since these systems typically operate in constrained, well-defined environments and thus usually exhibit predictable behavior. As a result, requirement analysis for such systems typically returns absolute answers, i.e., absolute success (requirements are satisfied by the system) or absolute failure (requirements are not satisfied by the system).

Non-traditional systems, on the other hand, such as ones that rely on Machine Learning (ML) components, bring uncertainty that impacts requirements specification and analysis. Requirements for ML components often use probabilities to quantify uncertainties about system behavior and the environment. Furthermore, their analysis may return either absolute or probabilistic answers.

Understanding the nature of requirements for systems that incorporate ML components is important, particularly if these systems are to operate in critical domains e.g., aerospace. To this end, we examine existing requirements from industry and academia to determine commonalities and differences between requirements for classical systems and those that learn. We discover that requirements for autonomous systems usually differ from classical systems because they make decisions based on learning and/or interactions with a stochastic physical environment. Specifically, these requirements often contain probabilities, e.g., "robot shall correctly identify rocks with 80% accuracy". Requirement specification tools like NASA's Formal Requirement Elicitation Tool (FRET) (Giannakopoulou et al. 2020) traditionally don't support writing requirements with probabilities. To this end, we are currently working on extending FRET to support probabilistic requirements in order to accurately express requirements for autonomous systems that can be used to guide the verification process and result in more reliable software.

In this talk, we present work that was previously published in (Farrell, Mavridou, and Schumann 2023). We also give a glimpse of ongoing work on FRET that has not been published yet.

Requirements for Autonomous Systems

We have gathered requirements from different sources. We only present a subset of these requirements (due to space limitations) in Tables 1and 2. First we present requirements that we gathered by performing a detailed literature review. Requirements **[LR-001]** and **[LR-002]** in Table 1 refer to the TaxiNet system (Frew et al. 2004), which uses a vision-based neural network to predict an aircraft's position on the runway relative to the center-line to enable autonomous runway taxiing. Table 1, Part 2 shows requirements that we elicited in conjunction with developers as part of the R-RAV project at NASA Ames, which takes a similar approach to TaxiNet and is driven by a neural network.

Table 2 contains 4 (out of 14) sanitized requirement patterns, which were obtained after manually analyzing 770 re-

AAAI 2024 Spring Symposium on User-Aligned Assessment of Adaptive AI Systems. Stanford University, Stanford, CA, USA.

Req ID	Requirement	Source
	Literature Studies	
[LR-001]	The aircraft location does not exceed a specified lateral offset from the runway centerline during taxiing.	(Asaadi et al. 2020; Păsăreanu et al. 2023)
[LR-002]	The aircraft does not veer off the sides of the runway during taxiing.	(Asaadi et al. 2020; Păsăreanu et al. 2023)
	R-RAV Project	
[RRAV-005]	Neural network shall output a sensible angle: the value must be between -90 and 90 degrees.	[R-RAV]
[RRAV-006]	The neural network shall achieve a minimum of X% accuracy on training and Y% accuracy on testing.	[R-RAV]
[RRAV-007]	(Local robustness) The neural network shall be robust to small perturbations in the image (pixels).	[R-RAV]

Table 1: Requirement examples from literature review and the NASA R-RAV project.

Req ID	Requirement Pattern (source: NASA)
[IC-001]	The sw shall achieve an average PARAMETER value of X.
[IC-002]	The sw shall estimate PARAMETER to within $+-X$ with a $Y\%$ confidence.
[IC-004]	The requirement shall be verified by measuring the average of the parameter over N repetitions.
[IC-012]	The sw shall detect CONDITION that implies EVENT is probable.

Table 2: Requirement patterns extracted from missions and industrial case studies.

quirements from missions and industrial case studies that use AI components. We call these *patterns* as they do not contain specific system details. Upper case variables must be instantiated by actual names and values to yield a multitude of similar requirements. Many of the requirements shown in Table 2 specify constraints on the computed parameters (e.g., **[IC-001]**) and have a notion of confidence (e.g., **[IC-002]**). The majority of the requirements that we collected use probabilities. Examples include: **[RRAV-006]** and **[IC-012]**.

We observed that notions of *confidence*, *accuracy*, and *average value* are often used to describe probabilistic requirements. In some cases, e.g., requirements **[LR-001]** and **[LR-002]**, although confidence levels are not part of the requirement text, they are added through separate requirement attributes. For example, **[LR-001]** must hold 95% of the time (lower confidence level), while **[LR-002]** must hold 100% of the time (always, higher confidence level), i.e., the TaxiNet system must avoid any runway excursion.

Extending FRET

FRET[1] is an open source framework developed at NASA Ames Research Center for capturing, understanding, formalizing, and analyzing requirements.

In practice, requirements are typically written in natural language, which is ambiguous and consequently not amenable to formal analysis. Since formal, mathematical notations are unintuitive, requirements in FRET are written in a structured natural language, named FRETish, which aims at providing a vocabulary natural to the user. Once a requirement is captured in FRETish, FRET automatically generates metric temporal logic (MTL) formulae that can be directly digested by formal analysis tools.

Traditionally, FRET does not support writing requirements with probabilities. To be able to capture and formalize such requirements, we are currently working on extending the FRETish language to support probabilistic requirements. A probabilistic FRETish requirement consists of six fields: 1) `condition` is a Boolean expression that whenever true specifies that the `response` shall happen; 2) `component` is the system component that the requirement is levied upon; 3) `shall` is used to express that the component's behavior must conform to the requirement; 4) `probability` defines the probability associated with the `timing` and `response`; 5) `timing` specifies when the response shall happen, subject to the constraints defined in `condition` and 6) `response` is the Boolean expression that the component's behavior must satisfy. An example requirement for TaxiNet is: `The aircraft shall with probability < 0.001 eventually satisfy turn > prescribedDegree`.

We are also building a formalization framework that supports the automatic translation of FRETish requirements into PCTL (Aziz et al. 1995) formulae. Our approach to formalization is compositional by leveraging instances of the FRETish fields explained above. The goal is to generate probabilistic formulae that can be directly digested and used for analysis by state-of-the-art probabilistic verification[2] and runtime monitoring tools.

[1] https://github.com/NASA-SW-VnV/fret

[2] E.g., for closed-loop analysis of vision-based autonomous systems (Păsăreanu et al. 2023)

References

Asaadi, E.; Beland, S.; Chen, A.; Denney, E.; Margineantu, D.; Moser, M.; Pai, G.; Paunicka, J.; Stuart, D.; and Yu, H. 2020. Assured Integration of Machine Learning-based Autonomy on Aviation Platforms. In *Digital Avionics Systems*, 1–10. IEEE.

Aziz, A.; Singhal, V.; Balarin, F.; Brayton, R. K.; and Sangiovanni-Vincentelli, A. L. 1995. It usually works: The temporal logic of stochastic systems. In *Computer Aided Verification: 7th International Conference, CAV'95 Liège, Belgium, July 3–5, 1995 Proceedings 7*, 155–165. Springer.

Farrell, M.; Mavridou, A.; and Schumann, J. 2023. Exploring Requirements for Software that Learns: A Research Preview. In Ferrari, A.; and Penzenstadler, B., eds., *Requirements Engineering: Foundation for Software Quality*, 179–188. Cham: Springer Nature Switzerland. ISBN 978-3-031-29786-1.

Frew, E.; McGee, T.; Kim, Z.; Xiao, X.; Jackson, S.; Morimoto, M.; Rathinam, S.; Padial, J.; and Sengupta, R. 2004. Vision-based road-following using a small autonomous aircraft. In *IEEE Aerospace Conference*, volume 5, 3006–3015.

Giannakopoulou, D.; Mavridou, A.; Rhein, J.; Pressburger, T.; Schumann, J.; and Shi, N. 2020. Formal requirements elicitation with FRET. In *Requirements Engineering: Foundation for Software Quality*.

Păsăreanu, C. S.; Mangal, R.; Gopinath, D.; Getir Yaman, S.; Imrie, C.; Calinescu, R.; and Yu, H. 2023. Closed-Loop Analysis of Vision-Based Autonomous Systems: A Case Study. In Enea, C.; and Lal, A., eds., *Computer Aided Verification*, 289–303. Cham: Springer Nature Switzerland. ISBN 978-3-031-37706-8.

Rierson, L. 2017. *Developing safety-critical software: a practical guide for aviation software and DO-178C compliance*. CRC Press.