Probably FRET?

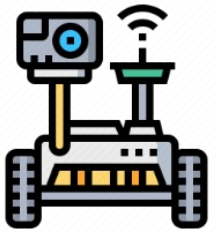
Anastasia Mavridou, Marie Farrell, Tom Pressburger, Johann Schumann

Probabilistic requirements?

- Reviewed requirements *from industry and academia*
- Let’s dive into probabilistic requirements and classify them into patterns

Probabilistic requirement patterns

"The probability of avoiding collisions with an obstacle shall be greater than 99% over the course of the mission"*



"With a probability of at least 0.95 no error will occur in the next 1000 seconds"

**this probability is application and system dependent*

Probabilistic requirement patterns

"The probability of avoiding collisions with an obstacle shall be greater than 99% over the course of the mission"

"With a probability of at least 0.95 no error will occur in the next 1000 seconds"

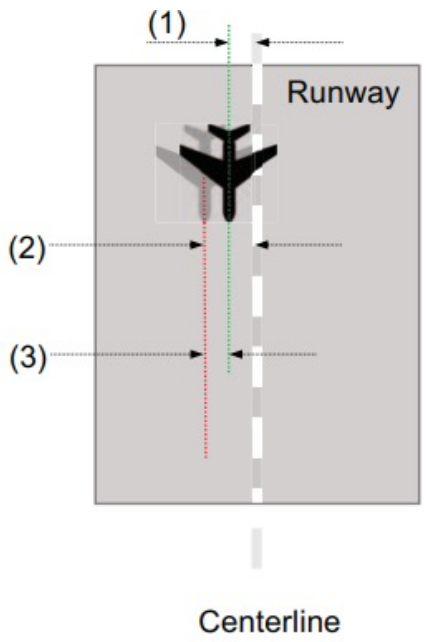
Probabilistic invariance

Predicate holds (always or continuously within a time bound) with a probability bound

Probabilistic requirement patterns

"The aircraft shall not leave the taxiway"

"The aircraft shall not turn more than a prescribed degree"



Corina Pasareanu, Ravi Mangal, Divya Gopinath, Sinem Getir Yaman, Calum, Imrie, Radu Calinescu, and Huafeng Yu,Closed-loop Analysis of Vision-based Autonomous Systems: A Case Study, Submitted to CAV 2023.

Erfan Asaadi, Ewen Denney, Ganesh Pai. Towards Quantification of Assurance for Learning-enabled Components, EDCC 2019.

Probabilistic requirement patterns

What is the probability that the aircraft eventually leaves the taxiway

What is the probability that the aircraft eventually turns more than a prescribed degree

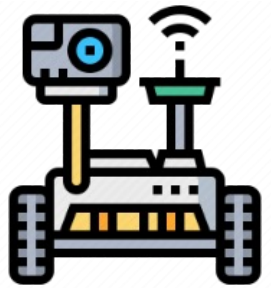
Calculate the probability that eventually the system reaches an error state

Probabilistic existence
Predicate will eventually become true with a probability bound

Probabilistic requirement patterns

“Whenever error X, the rover shall avoid collisions with a probability greater than bound”

“if FSM_STATE = MapSending_SendMapData, rover shall within <watchdog_timeout> transfer_map with probability greater than bound”



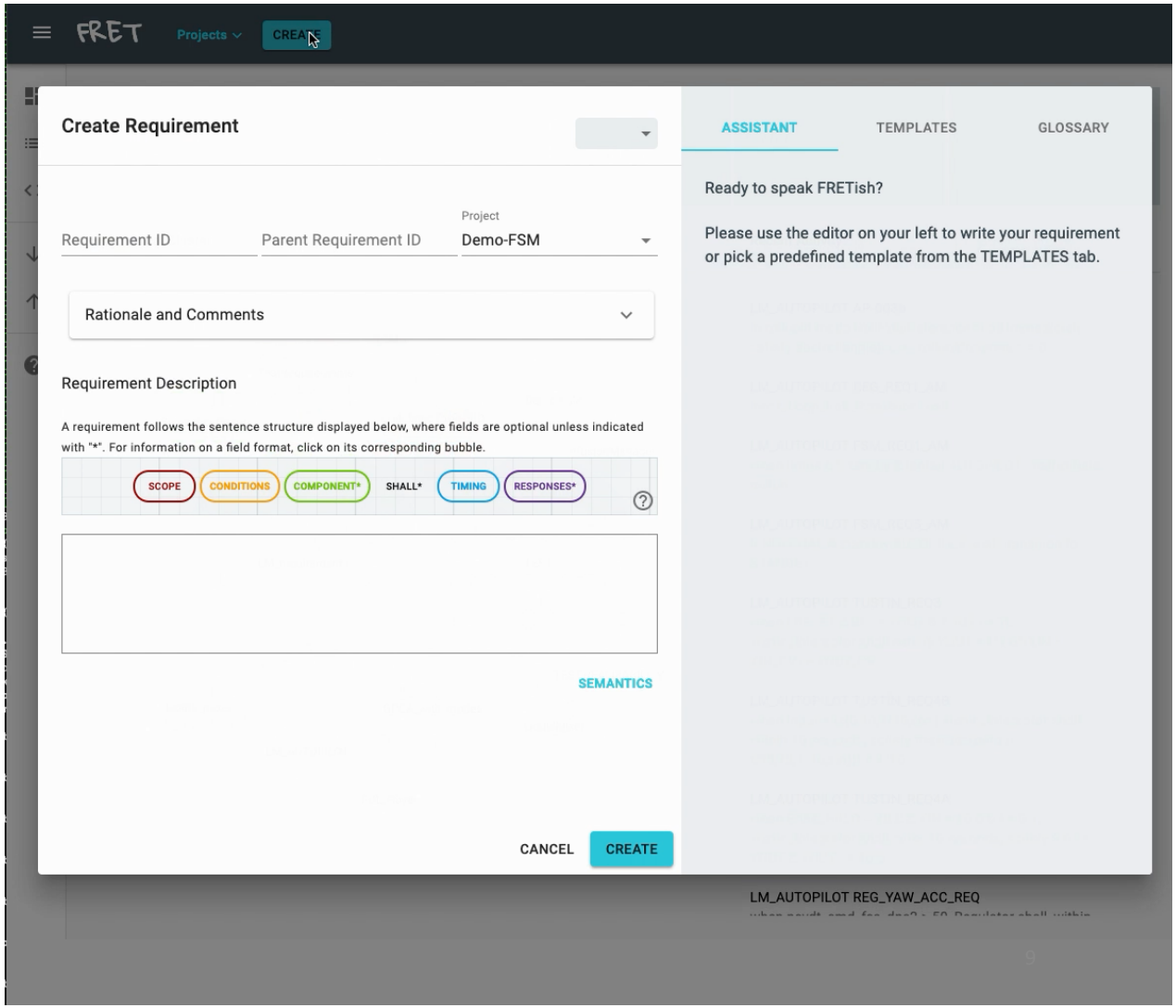
Probabilistic requirement patterns

"Whenever error X, the rover shall avoid collision with a probability greater than bound"

"if FSM_STATE = MapSending_SendMapData, rover shall within <watchdog_timeout> transfer_map with probability greater than bound"

Probabilistic response
Whenever predicate1 holds, predicate2 must become true with a probability bound

Do you speak FRETish?



Adding probabilities in FRET

template key: [scope, condition, timing]

SCOPE	null (global), in, before, after, notin, onlyIn, onlyBefore, onlyAfter
CONDITION	null, regular, trigger
TIMING	immediately, next, always, never, eventually, until, before, for, within, after

Adding probabilities in FRET

template key:[scope, condition, probability, timing]

SCOPE	null (global), in, before, after, notin, onlyIn, onlyBefore, onlyAfter
CONDITION	null, regular, trigger
PROBABILITY	null, bound, query
TIMING	immediately, next, always, never, eventually, until, before, for, within, after

Translate to the PRISM language

- PRISM's property specification language subsumes several probabilistic temporal logics
- Several probabilistic model checkers accept PRISM's language
  - E.g., STORM, ISCASmc

P operator: P bound [temporal property]

- P>0.99[F terminate]
- P>0.98[F (request & (X ack))]

P=? [property] quantitative approach

- P=?[F terminate]

Adding probabilities in FRET

[scope, condition, probability, timing]

For example:
[null, null, null, immediately]: P>=1[$response$]
[null, null, bound, immediately]: P~bound[$response$]
[null, null, query, immediately]: P=?[$response$]

Probabilistic invariance

[null, null, bound, always]: P~bound[G ($response$)]
[null, null, bound, for]: P~bound [G<=T ($response$)]

"The probability of avoiding collision with an obstacle shall be greater than 99% over the course of the mission"
FRETish: The rover shall with probability > 0.99 always satisfy ! collision
P>=0.99[G (! collision)]

"With a probability of at least 0.95 no error will occur in the next 1000 seconds."
FRETish: The sw shall with probability >= 0.95 for 1000 seconds satisfy ! error
P>=0.95[G<=1000 (! error)]

Probabilistic existence

[null, null, bound, eventually]: P~bound[F $response$]
[null, null, query, eventually]: P=?[F $response$]

"What is the probability that eventually the system reaches an error state"
FRETish: The aircraft shall with what probability eventually satisfy error

P=?[F error]

"What is the probability of battery depletion before completing mission objectives"
FRETish: The rover shall with what probability eventually satisfy battery_depleted

P=?[F battery_depleted]

Probabilistic response

[null, regular, bound, always]:
P>=1[G ($regular_condition$ -> P~bound[G $response$])]

"If error, the rover shall avoid collision with a probability greater than bound"
FRETish: Whenever error, the rover shall with probability > bound always satisfy ! collision

P>= 1[G (error => P>bound[G (! collision)])]

More complex formulas

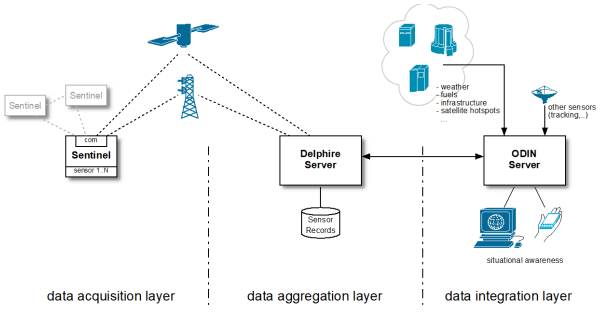
FRETish: if FSM_STATE = MapSending_SendMapData, rover shall with probability bound within <watchdog_timeout> satisfy transfer_map

P>=1[((G (((! (FSM_STATE = MapSending_SendMapData)) &
(X (FSM_STATE = MapSending_SendMapData))) =>
(P~bound[X (F<=watchdog_timeout transfer_map)]))) &
(FSM_STATE = MapSending_SendMapData) =>
P~bound[F<=watchdog_timeout transfer_map]))]

Ongoing work

- More than 40 different formalizations for probabilistic requirements
- Probabilistic requirements for ODIN-Fire

- *if ODIN gets a fire/smoke probability > X it sends notification within Y sec with a probability P_A*
- *if fire location > location threshold and fire spread vector > fire spread threshold the situational awareness system shall satisfy that the blockage probability for escape route 1 is P_BER1*



REFSQ 2023 paper

- ML requirement attributes and characteristics
- 13 sanitized requirement types
- Obtained by manually analyzing 770 requirements
- Missions and industrial case studies

Exploring Requirements for Software that Learns: A Research Preview

Marie Farrell[1], Anastasia Mavridou[2], and Johann Schumann[2]

[1] Department of Computer Science, The University of Manchester, UK
[2] KBR, NASA Ames Research Center, USA

Abstract. Context & motivation: 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. **Problem:** 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 especially difficult. **Principal ideas/results:** In this paper, we identify unique characteristics of software requirements that are specific to ML components. To this end, we collect and examine requirements from both academic and industrial sources. **Contribution:** To the best of our knowledge, this is the first work that presents real-life, industrial patterns of requirements for ML components. Furthermore, this paper identifies key characteristics and provides a foundation for developing a taxonomy of requirements for software that learns.

Bibliography

- M. Kwiatkowska, G. Norman, D. Parker, PRISM 2.0: A tool for probabilistic model checking, QEST, 2004.
- L. Grunske, Specification patterns for probabilistic quality properties. In ICSE, 2008.
- M. Autili, L. Grunske, M. Lumpe, P. Pelliccione, A. Tang, Aligning Qualitative, Real-Time, and Probabilistic Property Specification Patterns Using a Structured English Grammar. IEEE Transactions on Software Engineering, 2015.
- E. Asaadi, E. Denney, J. Menzies, G. J. Pai, and D. Petroff. Dynamic Assurance Cases: A Pathway to Trusted Autonomy. Computer, 2020.
- J. Horkoff. Non-functional requirements for machine learning: Challenges and new directions. In RE, 2019.
- D.M. Berry. Requirements engineering for artificial intelligence: What is a requirements specification for an artificial intelligence? In REFSQ, 2022.
- C. Menghi, C. Tsigkanos, M. Askarpour, P. Pelliccione, G. Vazquez, R. Calinescu, and S. Garcia. Mission Specification Patterns for Mobile Robots: Providing Support for Quantitative Properties. *IEEE Transactions on Software Engineering*. 2022.

Back-up slides

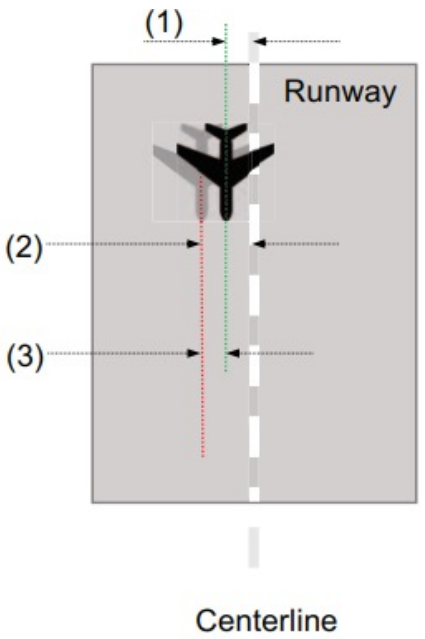
Derived TROUPE requirement

- For the entire mission*: The probability of collision with an obstacle shall be less than 1%*
- For a specific error (e.g., rover pose): *An error X is likely to occur with a probability 5%.*
- Derived requirement: *Given error X, rover shall always avoid collisions with a probability less than <calculatedValueToSatisfyMissionRqt>*

It implicitly defines the integrity of the barrier preventing or recovering from error X.

Probabilistic requirement patterns

"What is the probability of error conditioned on the model never aborting"



Conditional probabilities

What is the probability of formula1 conditioned on formula2

Conditional probabilities

"What is the probability of error conditioned on the model never aborting"
FRETish: Upon !Future(abort) the aircraft shall with what probability eventually satisfy error

P=?[F[error] & (! (F abort))] / P=? [! (F abort)]

Requirement Patterns

- 13 sanitized patterns
- Obtained by manually analyzing 770 requirements
- Missions and industrial case studies that use AI

“The sw shall estimate PARAMETER to be within +- X with a Y% confidence”