Formal Modelling and Analysis of a Self-Adaptive Robotic System

or On Modelling and Analysing Autonomous Underwater Robots as Probabilistic Featured Transition Systems

Maurice ter Beek

Consiglio Nazionale delle Ricerche

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"

FMT Formal Methods and Tools Lab

joint work with Juliane Päßler, Einar Broch Johnsen, S. Lizeth Tapia Tarifa (U Oslo)

Ferruccio Damiani (U Torino), and Clemens Dubslaff (TU Eindhoven)

IFIP WG 1.9/2.15 @ ISoLA'24, Crete, Greece

1 November 2024

REMARO RELIABLE AI FOR MARINE ROBOTICS

Funded by the European Union NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

Publications

- Formal Modelling and Analysis of a Self-Adaptive Robotic System (iFM'23)
  - Model: SAS from underwater robotics domain as *probabilistic FTS* with *dynamic feature switching*
  - Analysis: reward and safety properties with the *feature-aware probabilistic model checker* ProFeat
- A Configurable Software Model of a Self-Adaptive Robotic System (SCP'25)
  - Artefact published as *original software publication*
- Analysing Self-Adaptive Systems as Software Product Lines (JSS'25?)
  - Natural correspondence between SAS (*external self-adaptation* and *managed* + *managing* sub-system) and DSPL (*150% SPL family model* with a controller switching features during runtime)
  - Further adaptation: additional sensors with varying priorities, possible need to abort a mission
- Feature-Oriented Modelling and Analysis of a Self-Adaptive Robotic System (FAC'25?)
  - Further analyses using not only PRISM as backend but also Storm for *parametric model checking* and *multi-objective queries* (achievability, numerical, Pareto) depending on analysed properties:
    - Safety guarantees concerning mission duration and energy usage
    - Impact of different environments on the safety guarantees
    - Parameter synthesis to analyse the effect of environmental conditions
    - Trade-offs between mission duration and energy usage
    - …

https://pchrszon.github.io/profeat

https://www.prismmodelchecker.org/

https://www.stormchecker.org/

Problem

Analyse
Self-Adaptive Systems
as
Dynamic Software
Product Lines
with
ProFeat

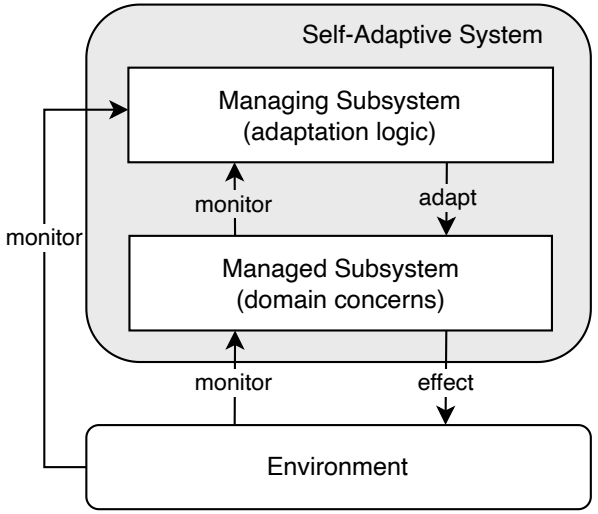
P. Chrszon, C. Dubslaff, S. Klüppelholz, C. Baier,
ProFeat: Feature-Oriented Engineering for Family-Based Probabilistic Model Checking.
Formal Aspects of Computing 30 (2018), 45–75.

Family-based model checking provides a means to *simultaneously* model check, in a *single* run, properties of a *family* of models, each representing a different *configuration*

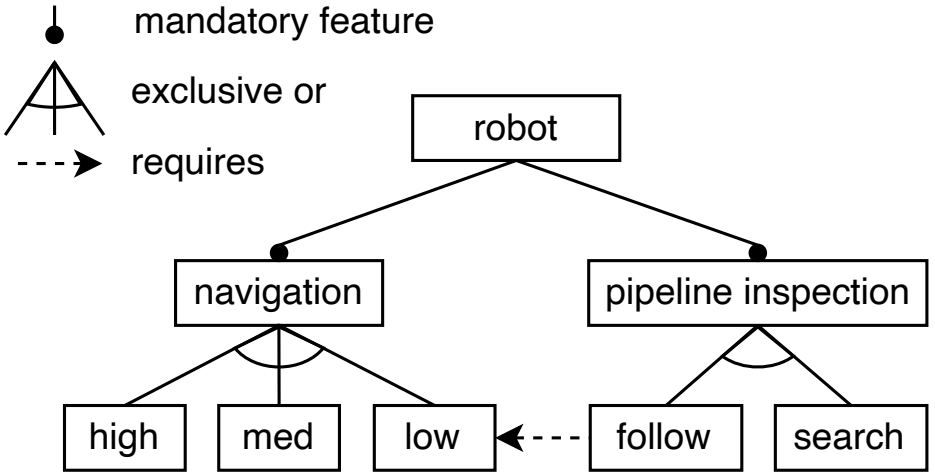
Pipeline Inspection

Formal Methods and Tools Lab

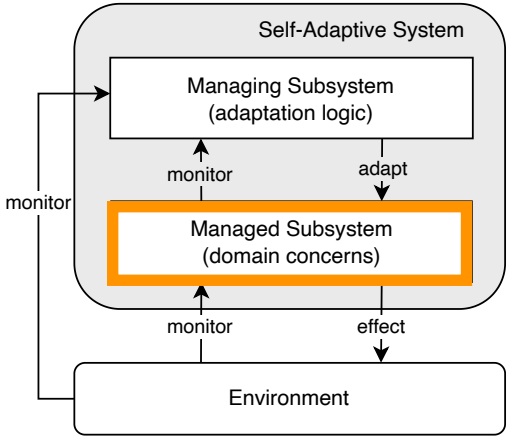
G. Rezende Silva, J. Päßler, J. Zwanepol, E. Alberts, S.L. Tapia Tarifa, Gerostathopoulos, E.B. Johnsen & C. Hernández Corbato, **SUAVE: An Exemplar for Self-Adaptive Underwater Vehicles**. *Proceedings of the 18th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2023)*, IEEE, 2023.

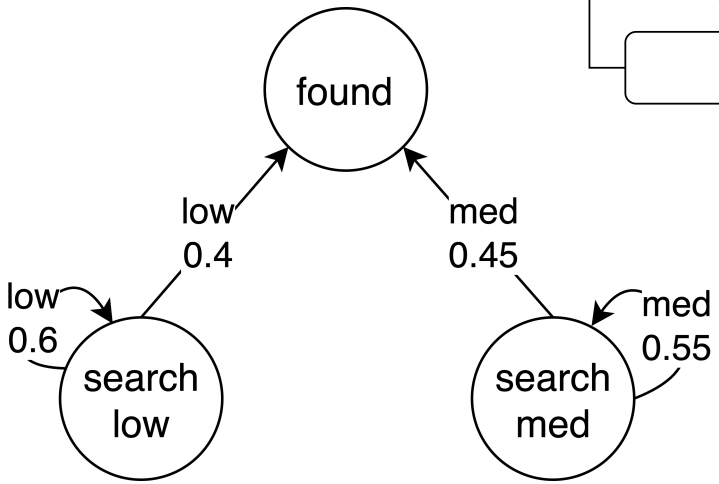


Probabilistic Featured Transition System



Feature Model

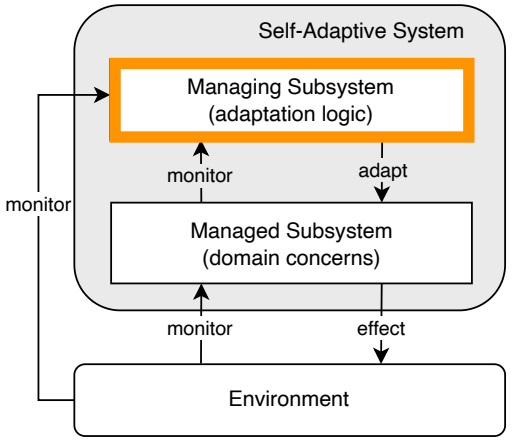




Probabilistic FTS

A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, J.-F. Raskin, **FTS: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking**. *IEEE Transactions on Software Engineering* 39 (2013)

Managing Subsystem



mandatory feature

exclusive or

requires

robot

navigation

pipeline inspection

high

med

low

follow

search

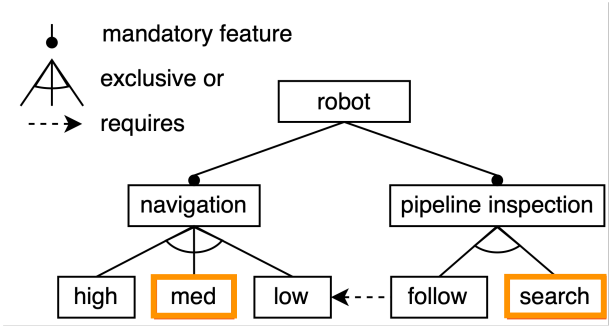
water visibility
medium

searching

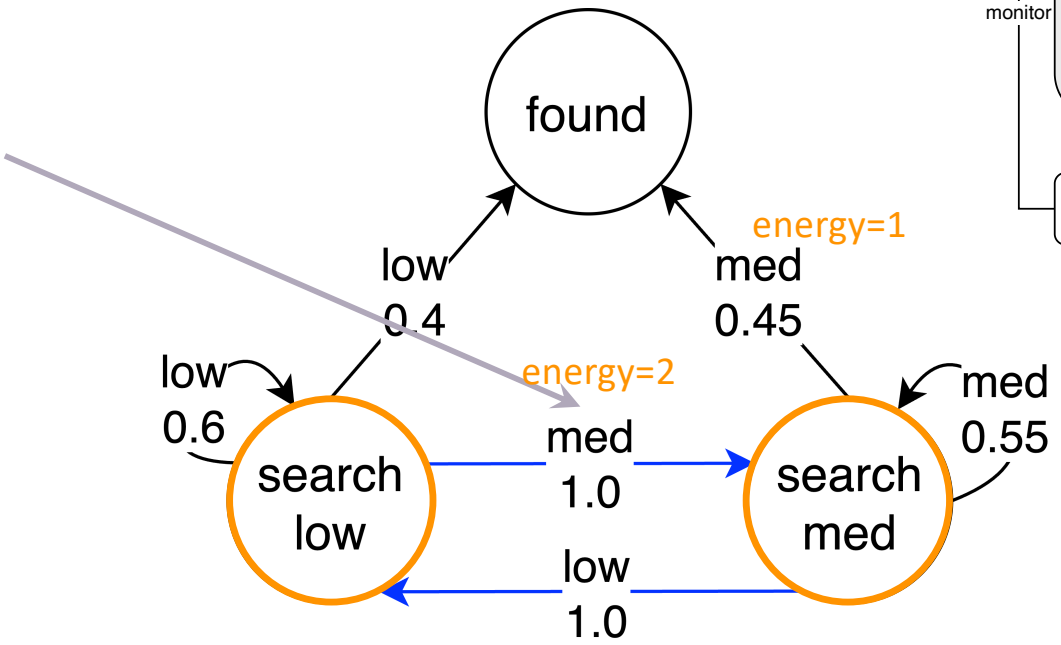
Environment

Managed Subsystem

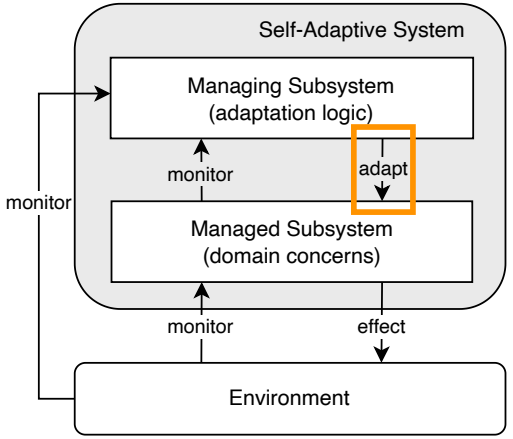
Adaptation



Managing Subsystem



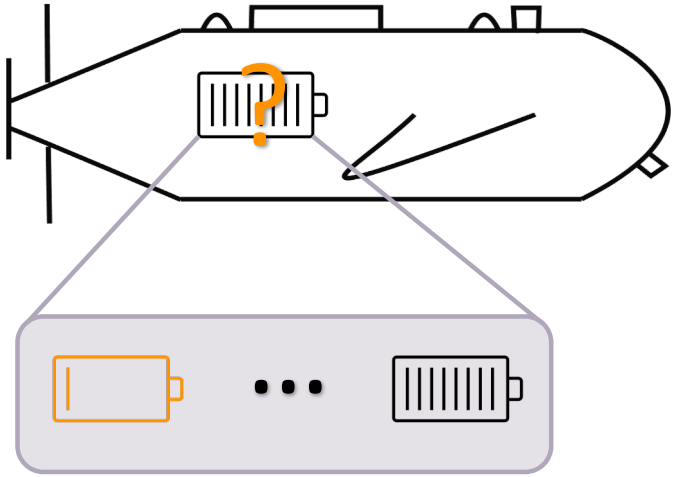
Managed Subsystem



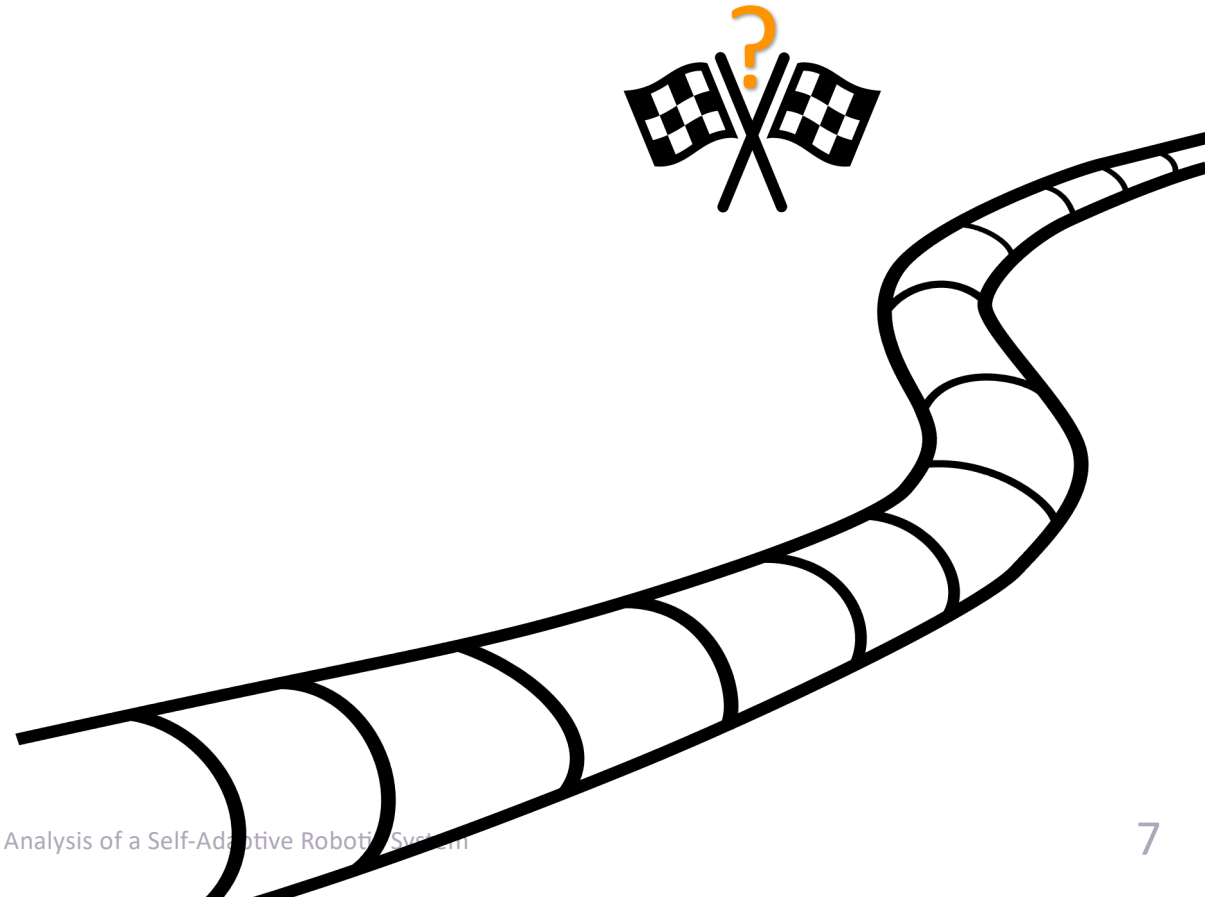
Analysis

Formal Methods and Tools Lab

Costs/Rewards



Probabilities



ProFeat Model of AUV

Formal Methods and Tools Lab

Costs/Rewards

```
root feature
  all of robot;
  modules auv, environment;
  rewards "time"
    [step] true : 1;
  endrewards
  rewards "energy"
    // Costs for being in a recovery state
    (s=recover_high) : 2;
    // .. omitted code ..

    // Costs for switching altitudes
    (s=search_high) & active(low) : 4;
    (s=search_high) & active(med) : 2;
    (s=found) & active(high) : 4;
    (s=found) & active(med) : 2;
    // .. omitted code ..
  endrewards
endfeature
```

Probabilities

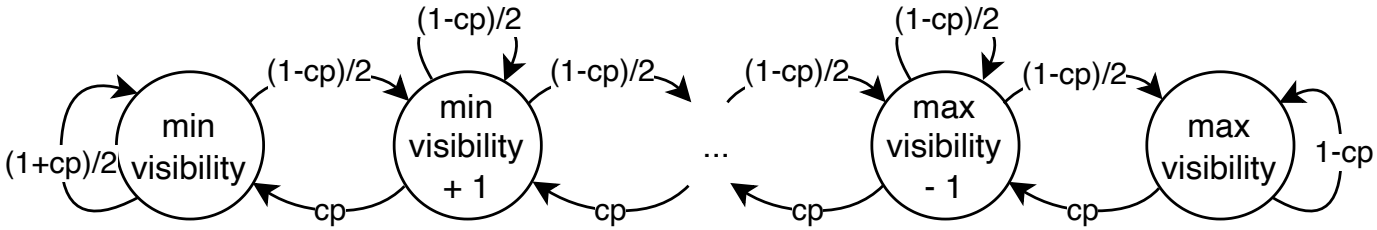
```
module auv
  s : [0..12] init start_task;
  d_insp : [0..inspect] init 0;
  t_failed : [0..infl_tf] init 0;

  // .. omitted code ..
  // From search state to another state
  [step] (s=search_high & active(high)) -> 0.59:(s'=found)
            + 0.4:(s'=search_high) + 0.01:(s'=recover_high);
  [step] (s=search_high & active(med)) -> 1:(s'=search_med);
  [step] (s=search_high & active(low)) -> 1:(s'=search_low);
  // .. omitted code ..

  // Following the pipeline
  [step] (s=following) & (d_insp<inspect) & (t_failed=0)
          -> 0.9: (s'=following) & (d_insp'=d_insp+1)
          + 0.07: (s'=lost_pipe) + 0.03:(s'=recover_following)
          & (t_failed'=(t_failed<infl_tf? t_failed+1 : t_failed));
  // .. omitted code ..
  [step] (s=following) & (d_insp=inspect) -> (s'=done);

  // Lost the pipeline
  [step] (s=lost_pipe) -> 1: (s'=start_task) & (t_failed'=0);
  // .. omitted code ..
endmodule
```

ProFeat Model of Environment




```
module environment
  water_visib : [min_visib..max_visib]
            init round((max_visib-min_visib)/2);
  [step] true -> current_prob: (water_visib'= (water_visib=min_visib?
            min_visib:water_visib-1)) + (1-current_prob)/2: (water_visib'=
            (water_visib=max_visib? max_visib:water_visib+1))
           + (1-current_prob)/2: true;
endmodule
```

ProFeat Feature Controller

Formal Methods and Tools Lab

```
1  formula med_visib = (max_visib-min_visib)/3;
2  formula high_visib = 2*(max_visib-min_visib)/3;
3
4  controller
5    // Change altitude depending on water visibility
6    [step] (s!=found) & active(search) & water_visib < med_visib
7           -> activate(low) & deactivate(high) & deactivate(med);
8    [step] (s!=found) & active(search)
9                & med_visib <= water_visib & water_visib < high_visib
10          -> activate(low) & deactivate(med) & deactivate(high);
11   [step] (s!=found) & active(search)
12               & med_visib <= water_visib & water_visib < high_visib
13          -> activate(med) & deactivate(low) & deactivate(high);
14   // .. omitted code ..
15
16   // Switch task from "search" to "follow"
17   [step] (s=found) & active(search)
18          -> deactivate(search) & activate(follow) & activate(low)
19               & deactivate(med) & deactivate(high);
20
21   // Switch task from "follow" to "search"
22   [step] (s=lost_pipe) & active(follow)
23          -> deactivate(follow) & activate(search);
24
25   // Enable transitions when following the pipeline
26   [step] (s!=lost_pipe) & active(follow) -> true;
27 endcontroller
```

ProFeat Analysis: Energy and Duration

Scenario	min_visib	max_visib	current_prob	inspect
1 (North Sea)	1	10	0.6	10
2 (Caribbean Sea)	3	20	0.3	30

```
1 R{"energy"}min=? [F ${s=done}];
2 R{"energy"}max=? [F ${s=done}];
```

	Energy		Time	
Scenario	min	max	min	max
1	24.78	44.39	23.66	32.40
2	59.08	4723.29	55.54	1315.58

ProFeat Analysis: Unsafe States

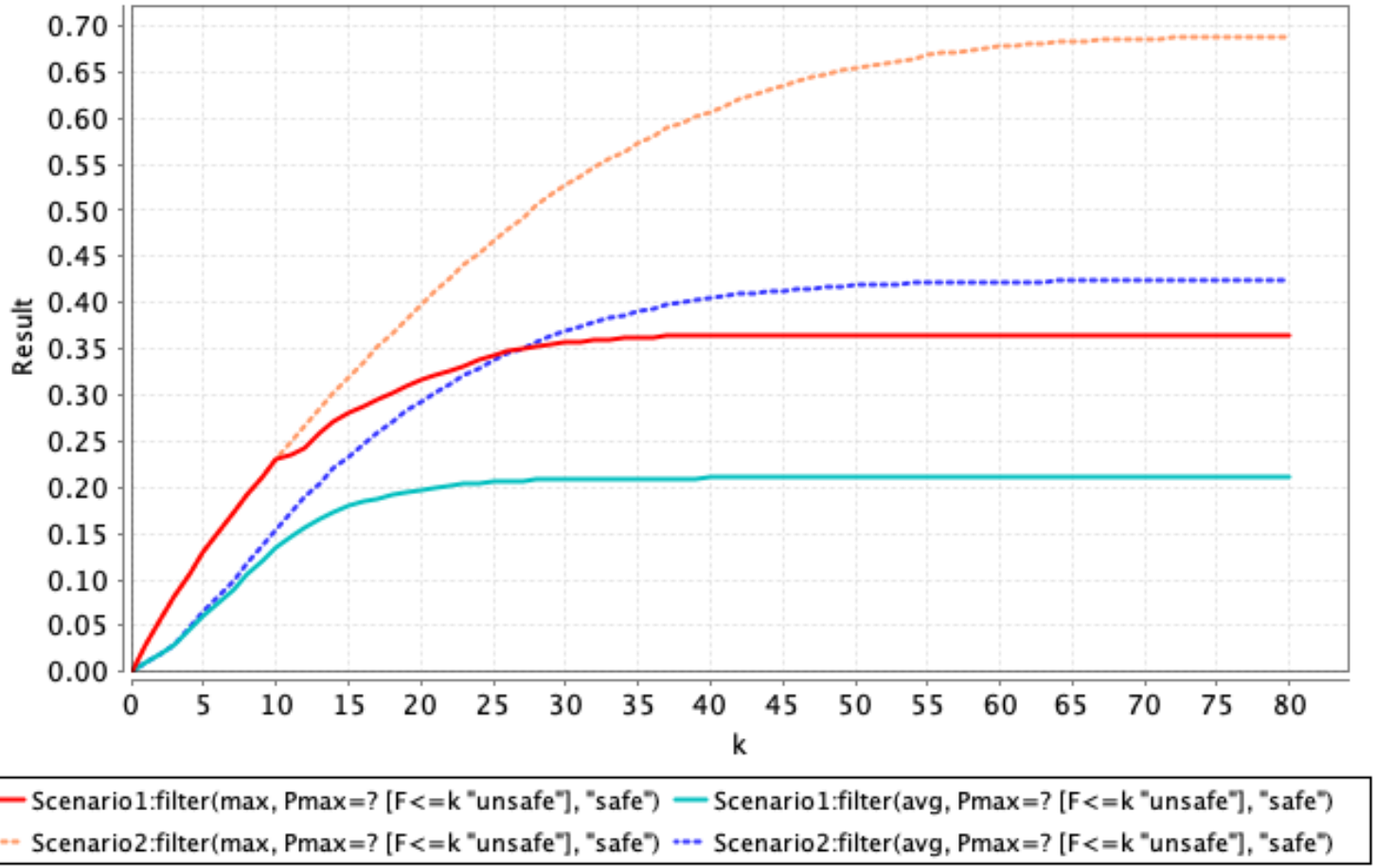
Formal Methods and Tools Lab

```
1 label "unsafe" = s=recover_high | s=recover_med | s=recover_low
2              | s=recover_following;
3 label "safe" = s=start_task | s=lost_pipe | s=start_search | s=search_high
4              | s=search_med | s=search_low | s=found | s=following |s=done;
5 Pmin=? [G "safe"];
6 filter(min, Pmin=? [ F<=k "safe" ], "unsafe");
7 filter(max, Pmax=? [ F<=k "unsafe" ], "safe");
8 filter(avg, Pmax=? [ F<=k "unsafe" ], "safe");
```

The minimum probability of only taking safe states (cf. Line 6) was shown to be 0.65 for Scenario 1 and 0.32 for Scenario 2.

In both of the scenarios, the probability of reaching a safe state from an unsafe state (cf. line 7) is above 0.95 after 5 time steps and above 0.99 after 7 time steps.

ProFeat Analysis: Unsafe States



Context

- *External* self-adaptation*:* separation of concerns between the application logic (the managed subsystem) and the adaptation logic (the managing subsystem) of the SAS
- *Bounded* adaptivity by modelling context variation that is anticipated at design time

- Dynamic SPLs have been proposed before to manage runtime reconfiguration for self-adaptive robots, yet it is still considered an unsolved challenge: "*there is a need to validate the proposals, either in an industrial environment or in different test cases, expanding the application areas*"

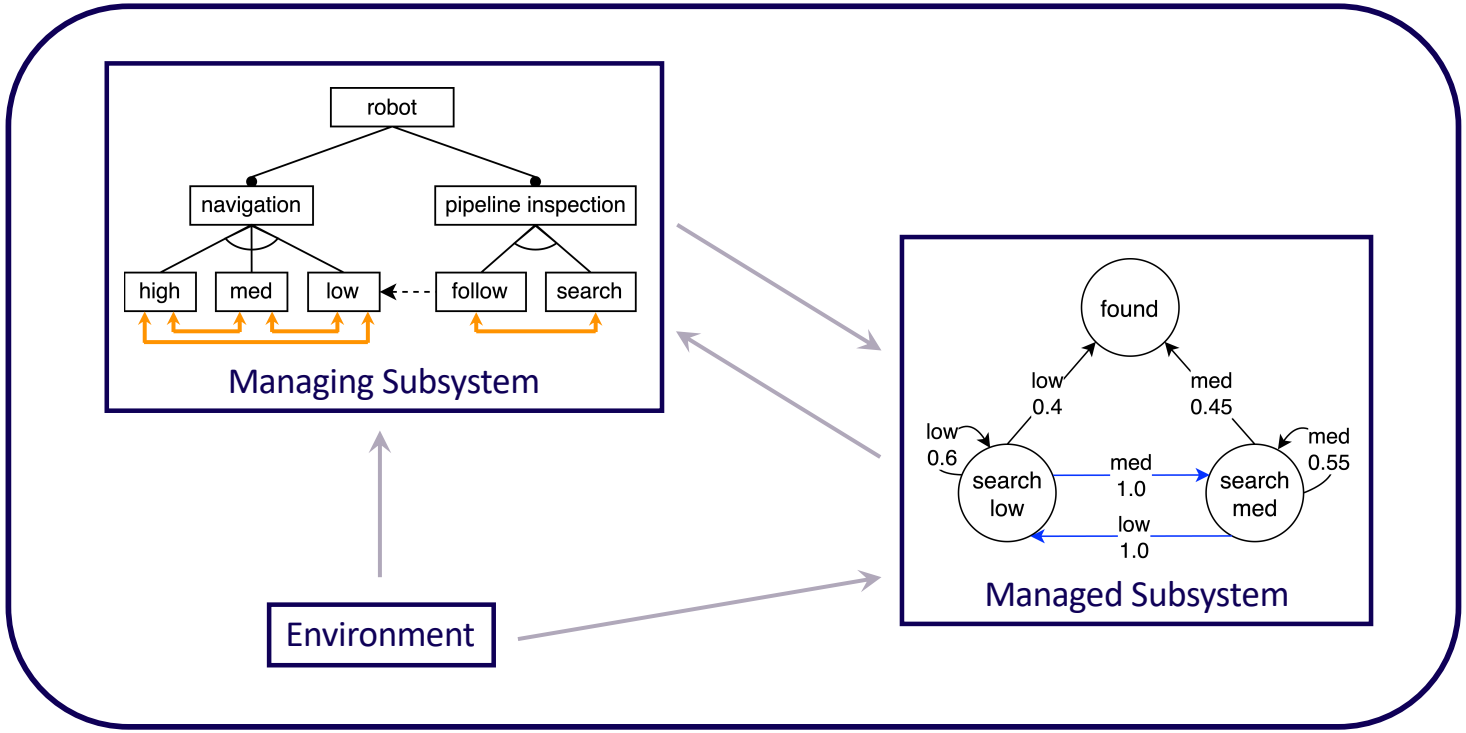
O. Aguayo and S. Sepúlveda, **Variability management in dynamic software product lines for self-adaptive systems—A systematic mapping**. *Applied Sciences* 12 (2022)

- K. Yoshimura, chief researcher at Hitachi, in his keynote address "*The 20-year journey of SPLE in Hitachi and the next*" during SPLC'23, presented the use of dynamic SPLs for autonomous robotic systems as a new industrial challenge

https://2023.splc.net/wp-content/uploads/2023/09/SPLC2023_Keynotes-1.pdf

Summary

Analyse SASs as DSPLs: ProFeat with family-based analysis



iFM'23 paper

SCP'25 paper

REMARO
RELIABLE AI FOR MARINE ROBOTICS
Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani