GRAN SASSO SCIENCE INSTITUTE

Democratizing the programming and use of Robots

Patrizio Pelliccione

Director of the Computer Science area

Full professor at GSSI

Adjunct professor at the University of Bergen, Norway

patrizio.pelliccione@gssi.it
https://www.patriziopelliccione.com/

www.gssi.it

Why Democratizing?

- **Accessibility:** technology (and robotics) extends to an ever-broader audience, and in some cases to the entire society
- **User-friendliness**: easier to use so that more people can use them (correctly and confidently) without needing advanced skills or training

Robots in hotels

- **Hotel concierge:** specifying what the robot should do
- **Hotel guests**: interact with the robots to give feedback or to make additional requests
- **Humans in the hotel:** share the environment with robots

Flyzoo Hotel - Alibaba Future Hotel Hangzhou

https://flyzoo-hotel.hangzhouhotel.org/en/

Autonomous car

- **Ride specification:** more complex than just specify the destination address
- **Degree of automation**: partial automation can be more stressful than fully manual driving, as drivers need to constantly monitor whether the vehicle is doing what it is supposed to

Industrial Robots

- **Experts in satellite production:** specifying what the robot should do
- **Customization in the production islands:** self-contained, flexible manufacturing unit with its own specificity that operates independently while still integrating with the larger smart factory ecosystem

PRESS RELEASES

Thales Alenia Space unveils project to develop Space Smart Factory, one of the largest facilities of its kind in Europe

Available in IT FR ES IMG PDF

NOV 9 2023

At the Tecnopolo Tiburtino hub in Rome, Thales Alenia Space's all-digital factory will employ advanced technologies for the production of satellites

- *The factory will be built thanks to an important investment by Thales Alenia Space and co-funded by the Italian Space Agency (ASI) through the National Recovery and Resilience Plan (PNRR) funds*
- *It will make intensive use of digital and Industry 4.0 technologies*
- *The factory will feature the Space JOINTLAB, an innovative and collaborative space with SMEs and research centers*
- *Total surface area 21,000 sq.m, 5,000 sq.m of reconfigurable clean rooms, 1,900 sq.m of office space and co-working areas, 1,800 sq.m of technical support areas*

https://www.thalesaleniaspace.com/en/press-releases/thales-alenia-space-unveils-project-develop-space-smart-factory-one-largest

Why democratization and not just user friendliness?

Different degree and complexity of interaction that get close to programming

Robotic Mission

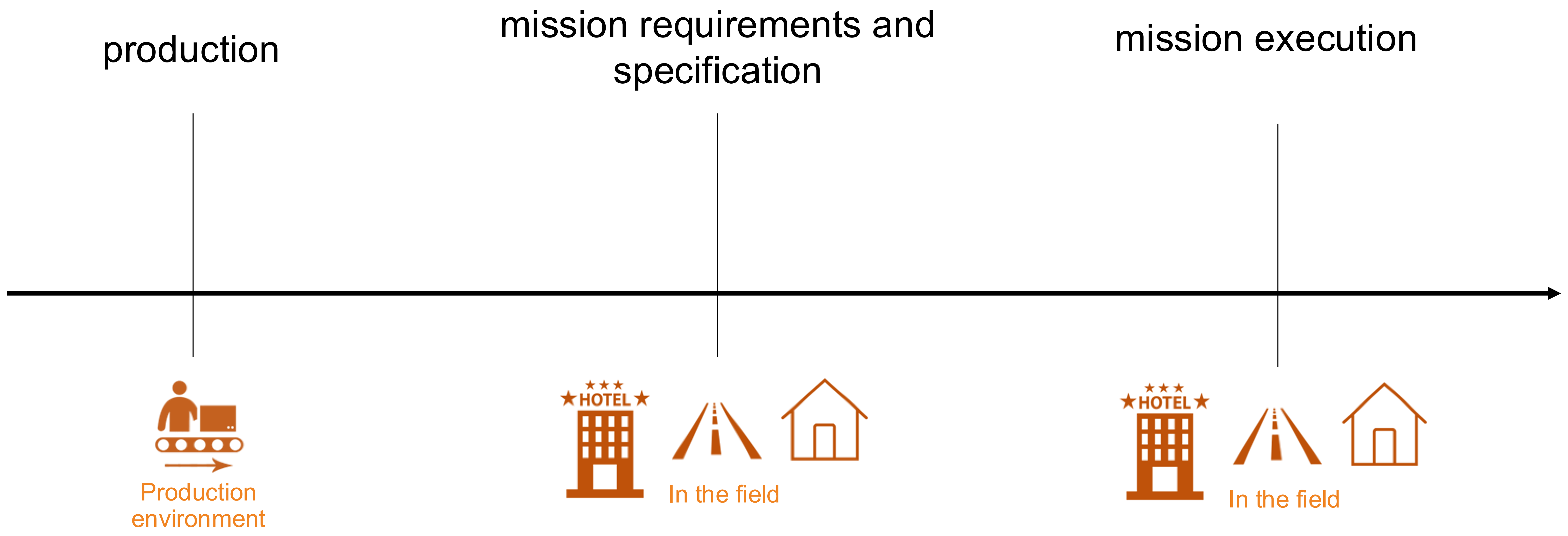
- A mission requirement describes the high-level tasks that a robotic software must accomplish.

- A mission specification is a formal and precise description of what robots should do in terms of movements and actions.

- Robotic mission engineering concerns expressing robotic missions in high-level and user-friendly notation (mission requirements), and then translating mission requirements into more precise mission specifications.

Claudio Menghi, Christos Tsigkanos, Patrizio Pelliccione, Carlo Ghezzi, and Thorsten Berger, Specification Patterns for Robotic Missions, Transactions on Software Engineering (TSE), 2019

Robots programming: beyond production environment

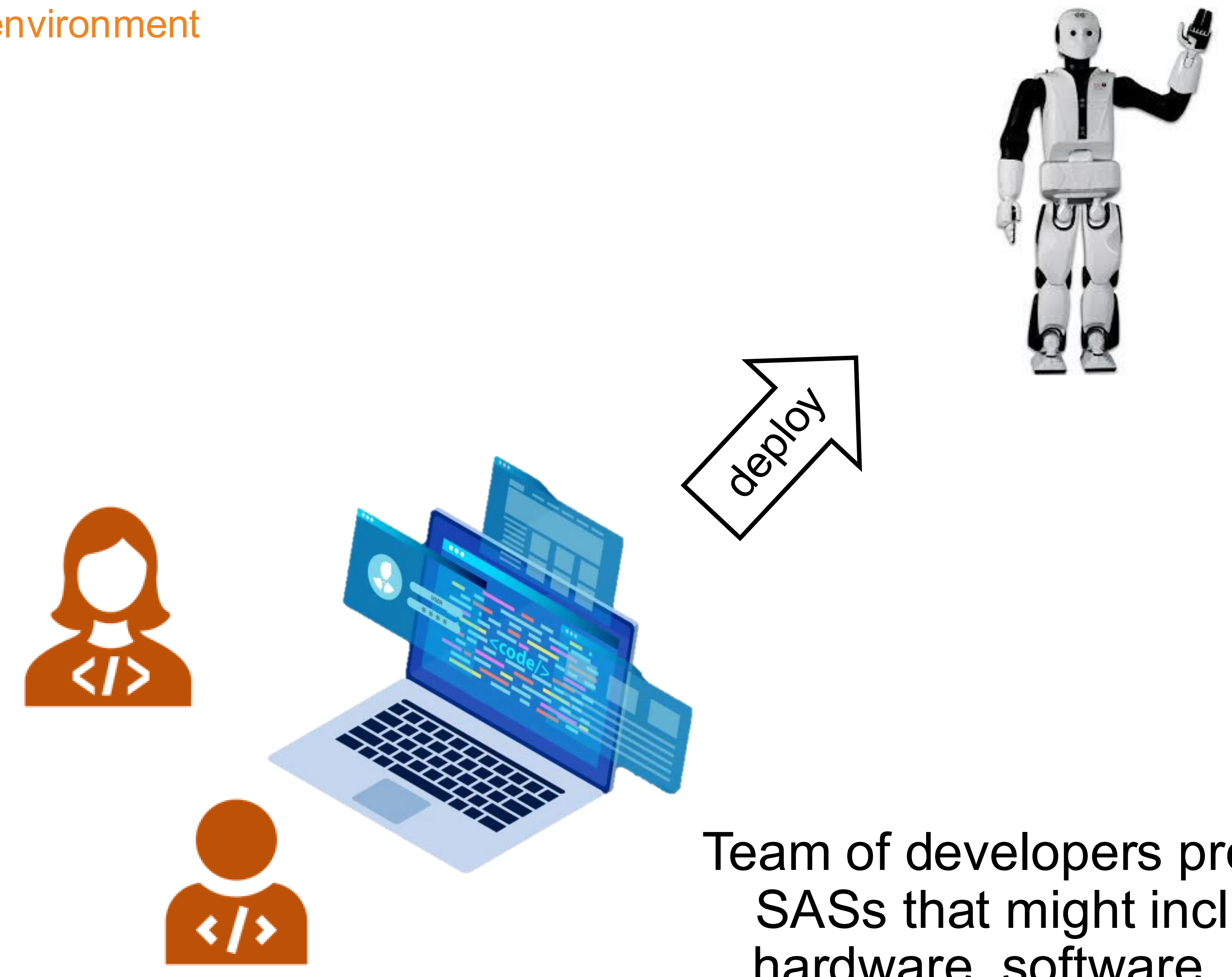


Different stakeholders

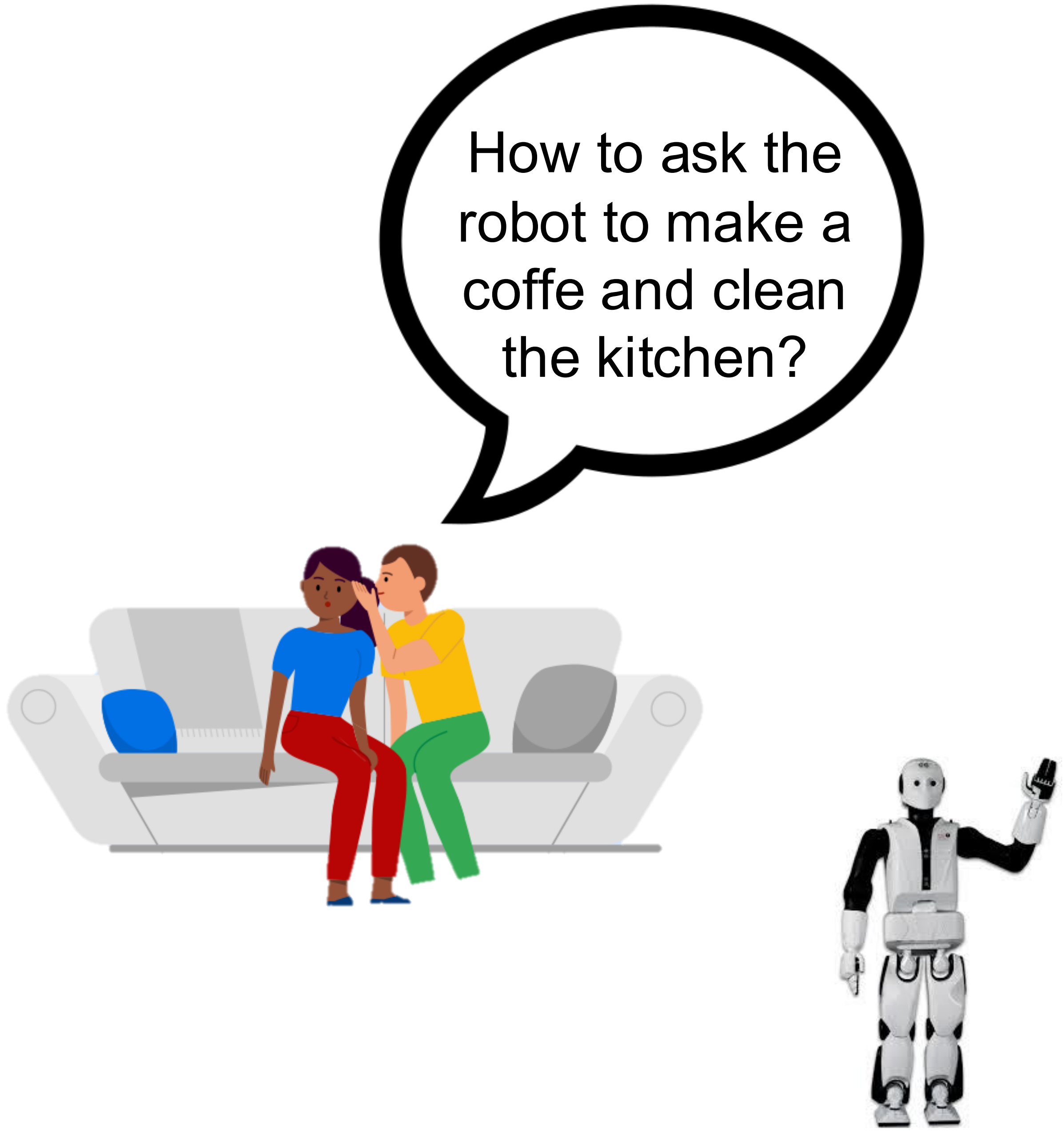
Mission Specification

Production environment

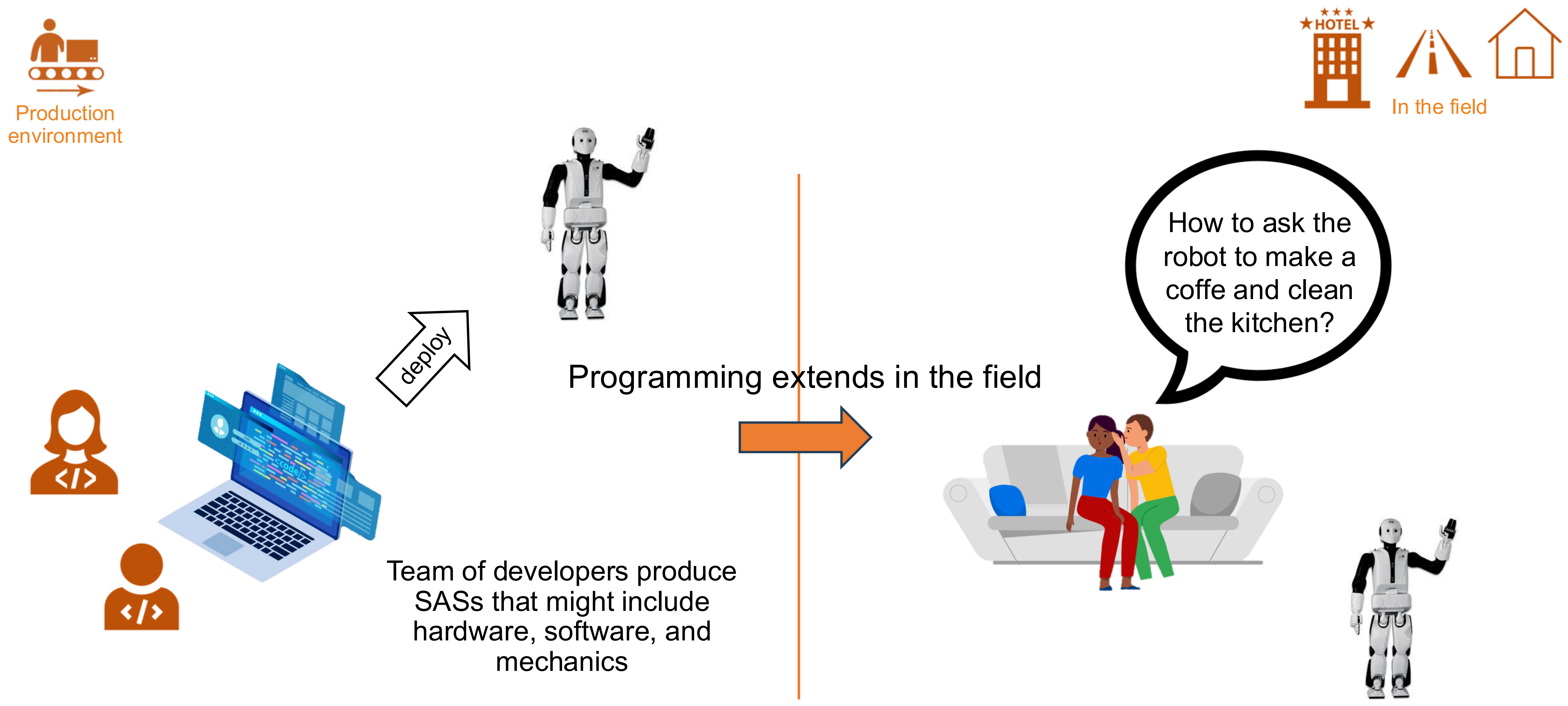
In the field



Team of developers produce SASs that might include hardware, software, and mechanics



Mission Specification



Need of Turn-key solutions

The definition of the mission should be done in an easy and user-friendly way, accessible by users without expertise in ICT or robotic

Different stakeholders, experts of the domain but not in robotics

Example of mission requirement

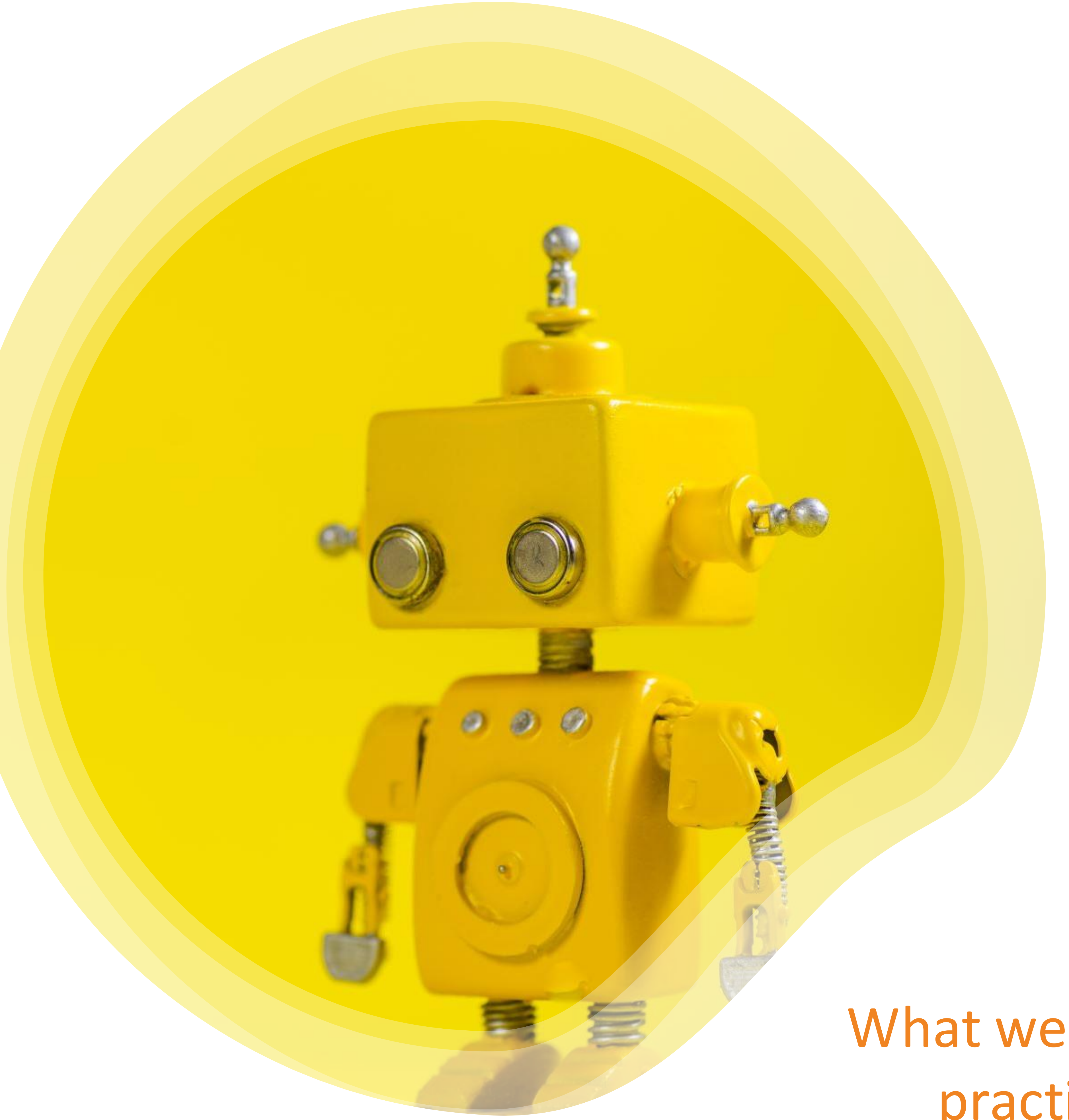
Different modes, plus dealing with the variability of the real world

The robot should **move around** the room and **dispense medication** to independent people. It first **establishes a short conversation** based on the user's conditions to figure out the overall health status, and afterwards it will **dispense pills** along with a glass of water. The robot also **records the activity** to allow a caregiver to evaluate if the persons accepted the pills, by means of a subsequent interaction. **During night-time** a service robot performs a **cleaning protocol with the UV-lamp** on the exposed surfaces (e.g. table and chairs), possibly in **coordination with automatic cleaners** that wash the floor.

In addition, the robot should **perform regular check-ups** on people with particular conditions during free-time. It **needs to understand basic requests** and will alert the nurse in case of need, pose basic riddles or show simple pictures to test baseline human capabilities, ask the persons about their status and if they need help or assistance. Through specific questionnaires the **robot gives advices about common pathologies** affecting elderly people such as heart failure or diabetes. Tasks in hospitals are very specialized and follow very strict protocols. For these reasons, as well as efficiency, patients might stay alone for long periods of time, causing them distress and confusion. Children are a particularly affected group, as their attachment to parents is high and it is difficult for them to understand the situation.

+ recharge battery when needed, deal with obstacles, presence of humans, failures, etc.

Exemplars: https://github.com/Askarpour/RoboMAX
Video: https://www.youtube.com/watch?v=txZCcABkycQ



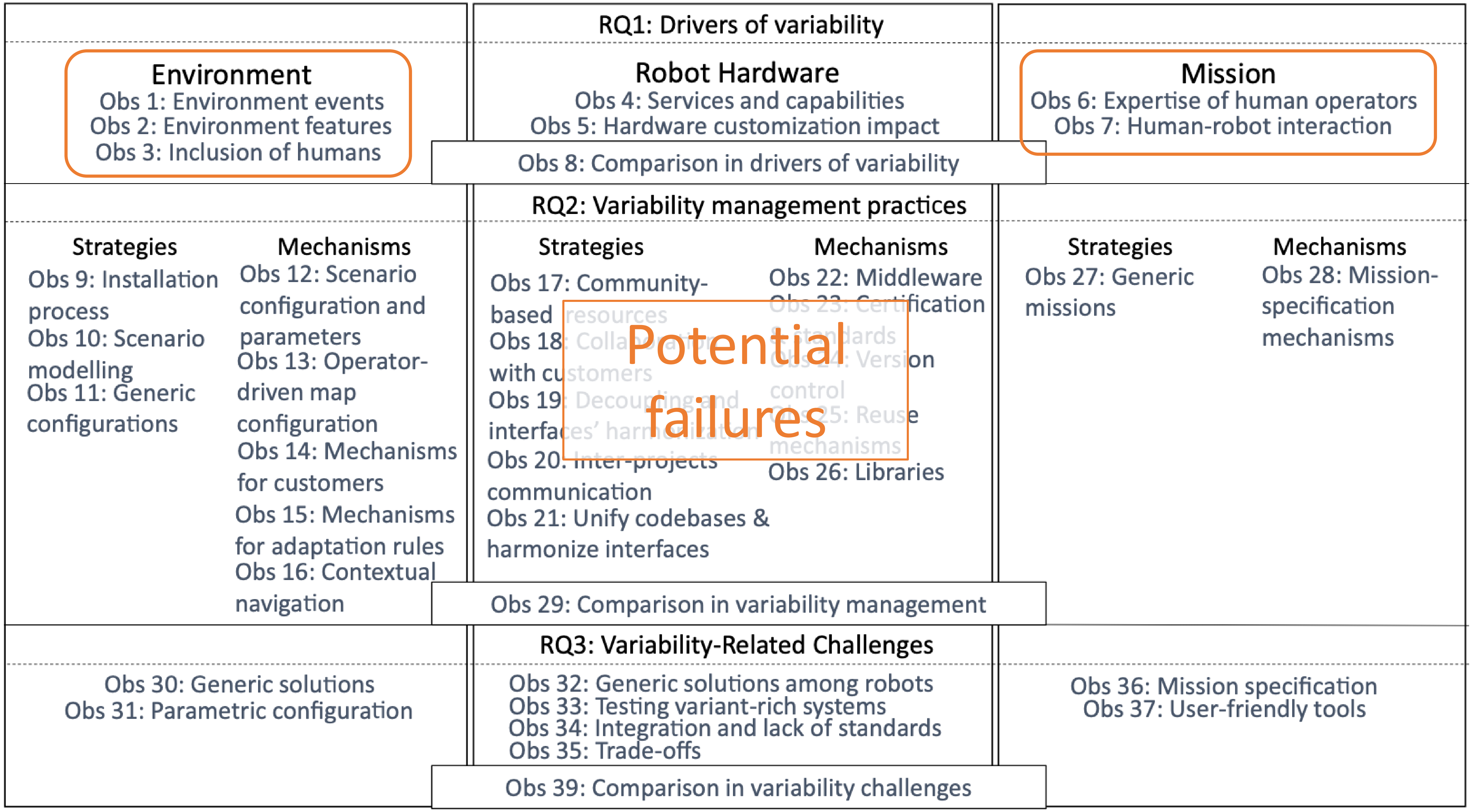
Variability and uncertainty

It’s not difficult to specify what the robot should do, i.e., the “normal” behavior.

The difficult part is to deal with uncertainty and exceptional behaviors while guaranteeing safety and the mission satisfaction.

What we learn from practitioners

Which type of variability?



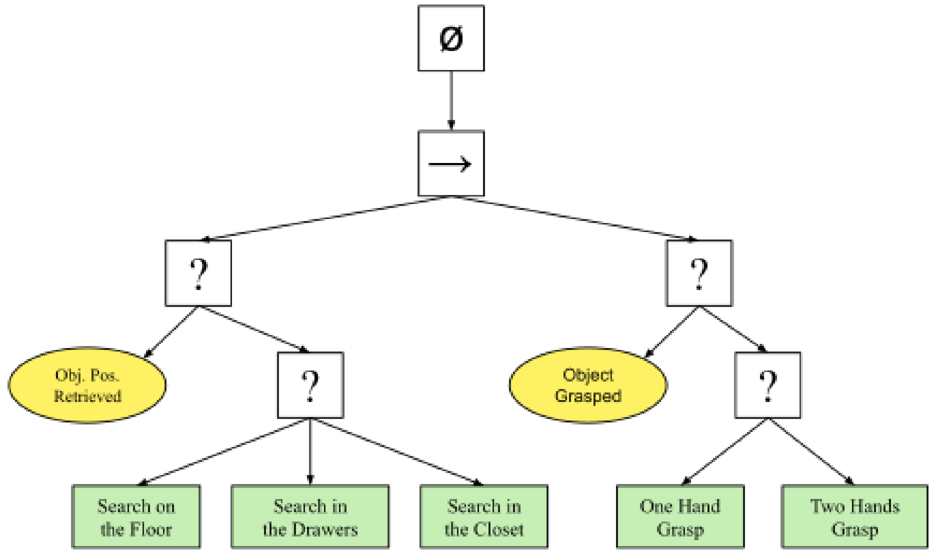
Sergio García, Daniel Strüber, Davide Brugali, Alessandro Di Fava, Patrizio Pelliccione, and Thorsten Berger. 2022. Software variability in service robotics. Empirical Softw. Engg. 28, 2 (Mar 2023). https://doi.org/10.1007/s10664-022-10231-5

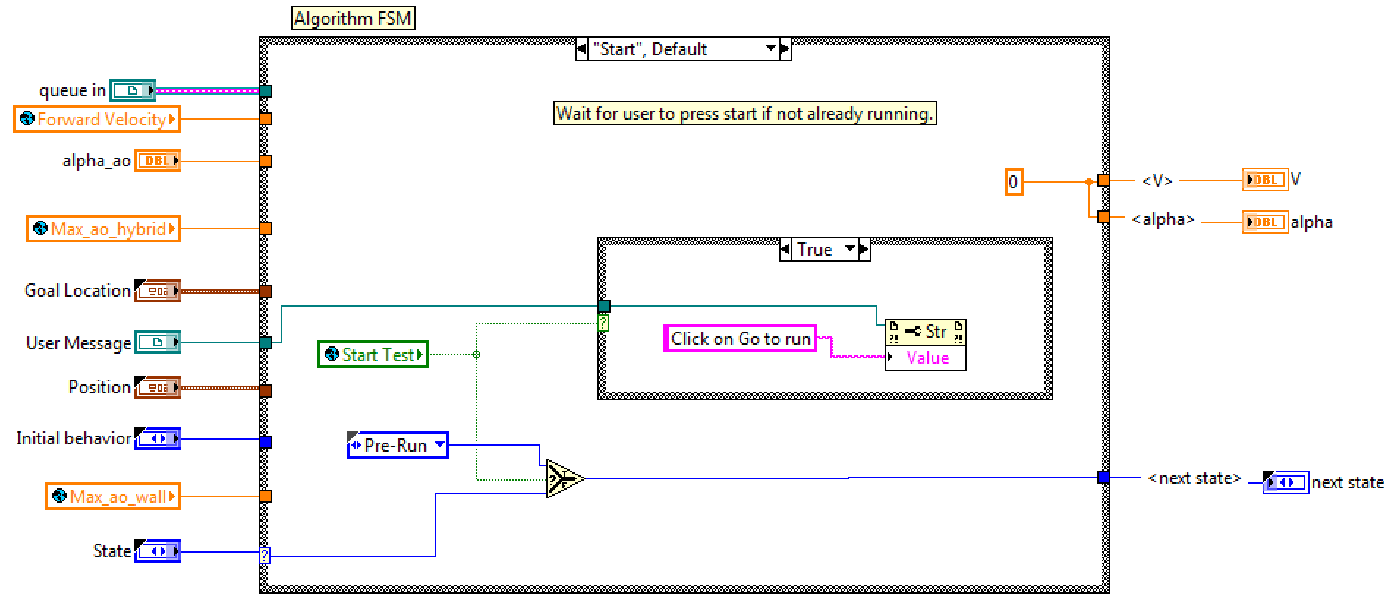
1. Democratization
2. Need of Turn-key solutions
3. Variability of the real world

How to specify missions?

GSSI

$\Phi1=<>((r \text{ in } l1) \&\& <>(r \text{ in } l2))$



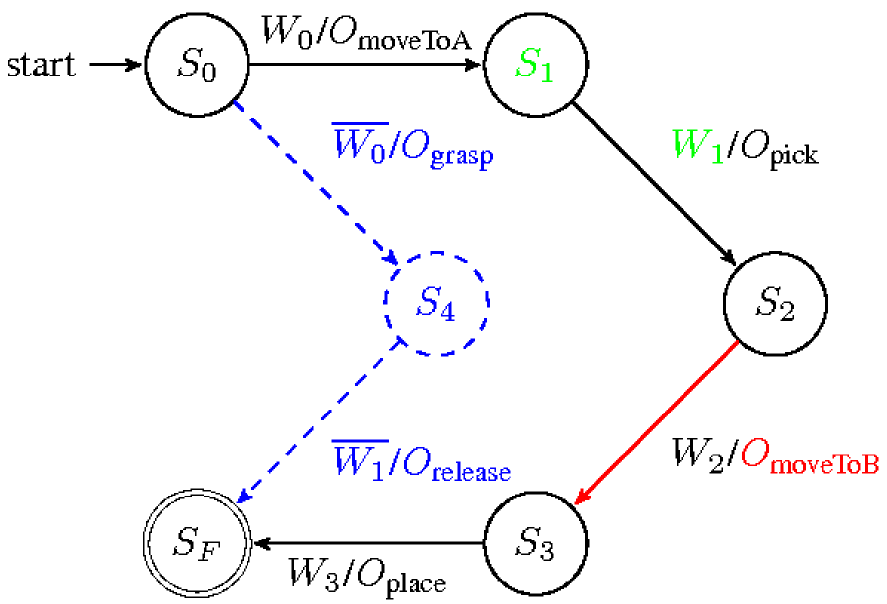



```
# Move the robot to the reference point:
robot.MoveJ(target)

# Draw a hexagon around the reference target:
for i in range(7):
    ang = i*2*pi/6 #ang = 0, 60, 120, ..., 360

    # Calculate the new position around the reference:
    x = xyz_ref[0] + R*cos(ang) # new X coordinate
    y = xyz_ref[1] + R*sin(ang) # new Y coordinate
    z = xyz_ref[2]              # new Z coordinate
    target_pos.setPos([x,y,z])

    # Move to the new target:
    robot.MoveL(target_pos)
```



Simple enough?

Means to specify robotic missions

Temporal logic: Φ1=<>((r in l1) && <>(r in l2))

Logic-based specification of robotic missions

- Pros:
  - Clear semantics and unambiguous specification
  - Can be directly used by planners to generate plans or synthesis approaches to generate controllers
  - Enable automatic verification
- Cons:
  - Require specific competencies and error-prone
  - Impossible or difficult to specify missions that are complex and with high variability

Logic-based specification of robotic missions

The logic of bugs

Author: Gerard J. Holzmann | Authors Info & Claims

SIGSOFT '02/FSE-10: Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering
Pages 81 - 87 • https://doi.org/10.1145/587051.587064

Published: 18 November 2002 Publication History

Abstract

Real-life bugs are successful because of their unfailing ability to adapt. In particular this applies to their ability to adapt to strategies that are meant to eradicate them as a species. Software bugs have some of these same traits. We will discuss these traits, and consider what we can do about them.

How to make logic more accessible and user friendly?

Let’s take inspiration from the formal verification world

Problem space

- Temporal Properties are typically specified as formulae in suitable temporal logics
- The inherent complexity of Temporal Logic formulae may induce to specify properties in a wrong way

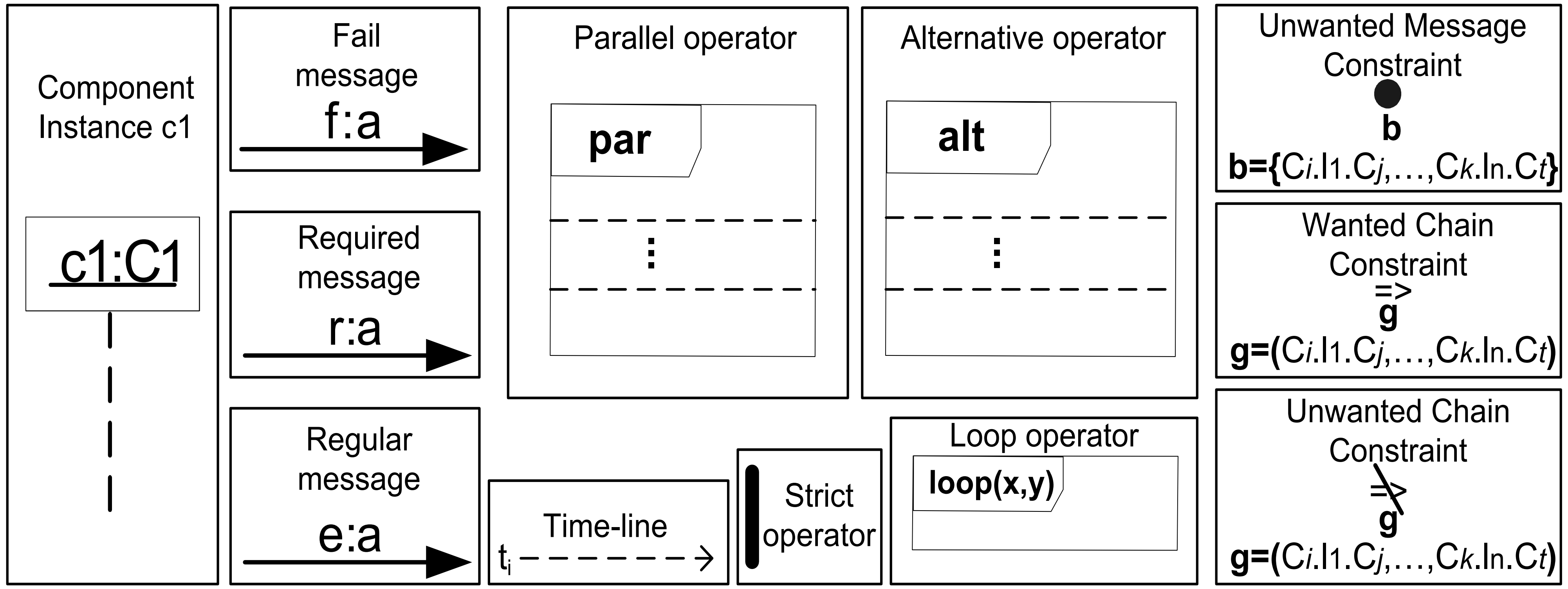
Solution space

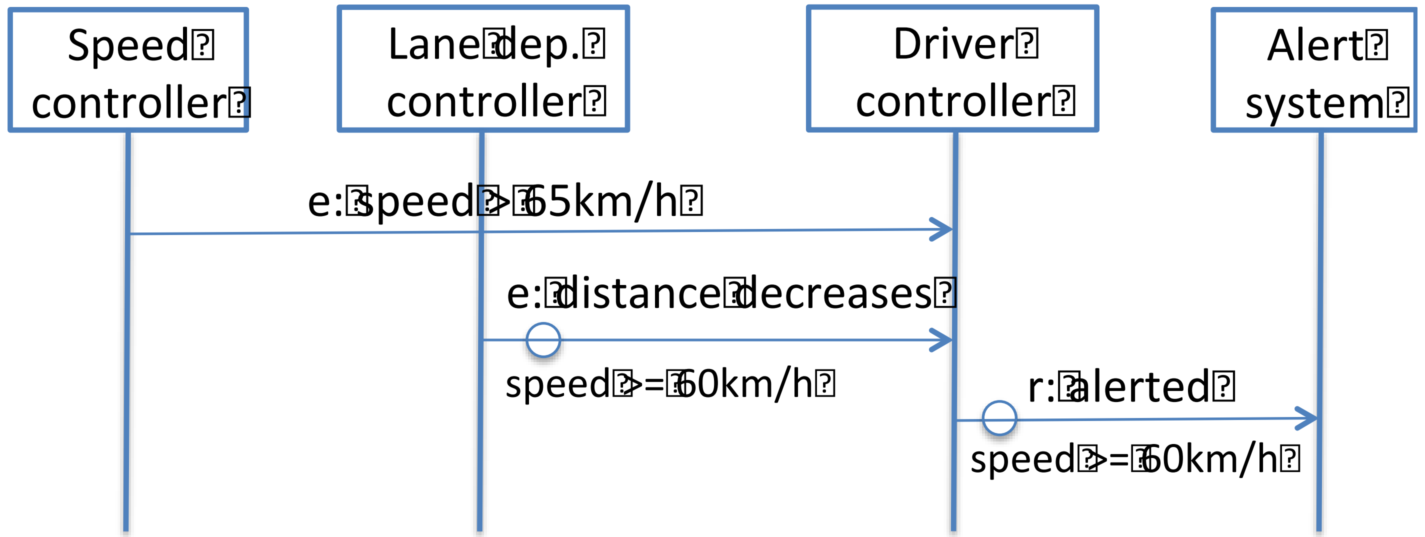
- Languages to facilitate the temporal properties specification
- Property Specification Patterns

Main idea

Reduce the expressivity to what is really needed and simplify

Properties Sequence Chart (PSC)





- Extensions and uses of PSC
  - Timed Property Sequence Chart (TPSC) - http://dx.doi.org/10.1016/j.jss.2009.09.013
  - Probabilistic Timed Property Sequence Chart (PTPSC) - http://dx.doi.org/10.1109/ASE.2009.56
  - Monitoring of PSC and TPSC properties - http://dx.doi.org/10.1007/978-3-642-16612-9_39
  - Monitoring of PTPSC - http://onlinelibrary.wiley.com/doi/10.1002/spe.1038/abstract

PSC is one of the notations adopted within the Presto project (ARTEMIS-2010-1-269362)
http://www.presto-embedded.eu/

PSC is the notation used by MSC Tracer to express temporal properties
http://www.pragmadev.com/product/tracing.html

PSC is the notation used by SDL-RT V2.3 standard to express temporal properties
http://www.sdl-rt.org/

Marco Autili, Paola Inverardi, Patrizio Pelliccione (2007) **Graphical scenarios for specifying temporal properties: an automated approach** , Automated Software Engg. 14: 3. 293-340 http://www.di.univaq.it/psc/

Property specification patterns

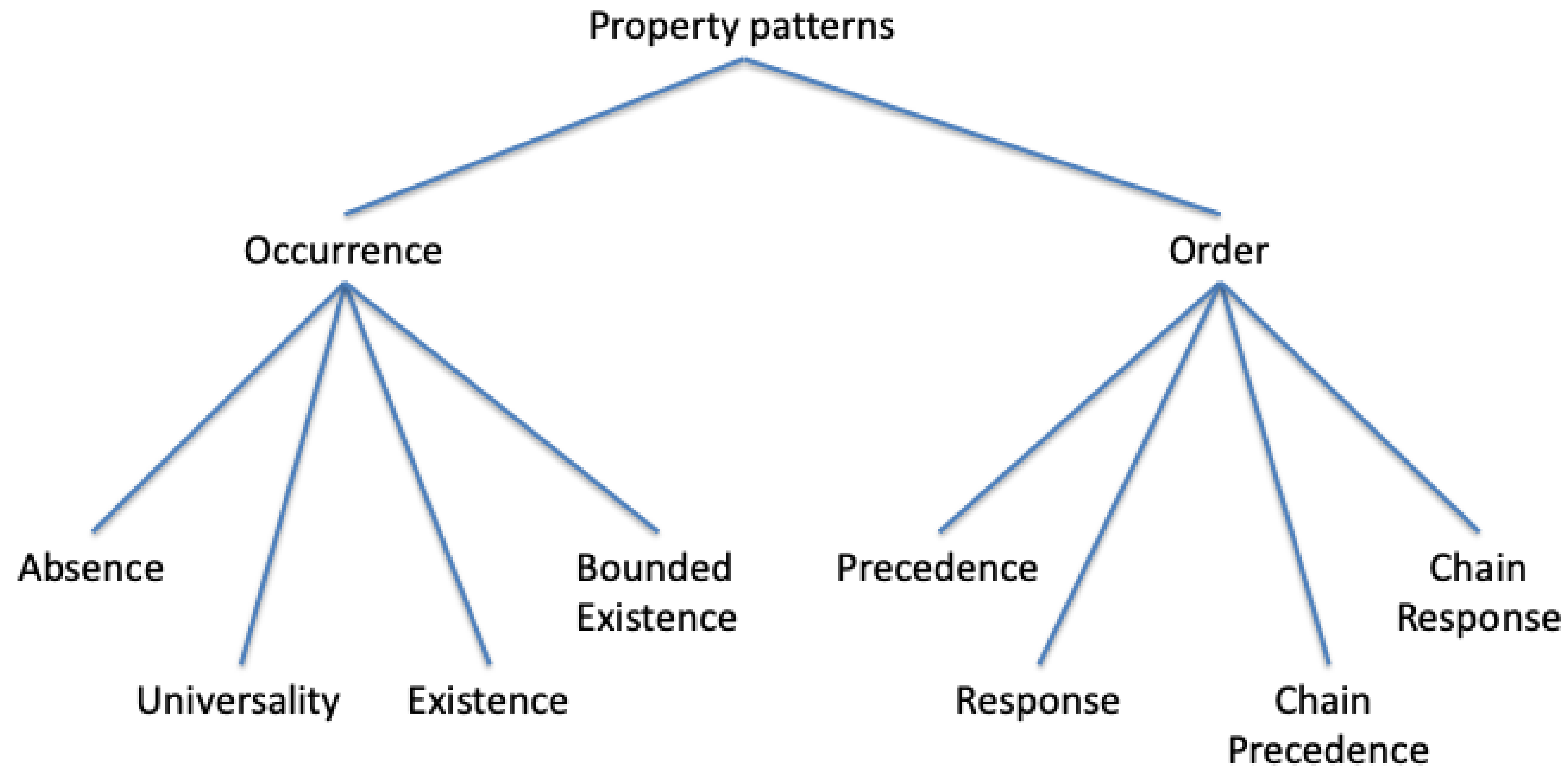
An example: Response pattern

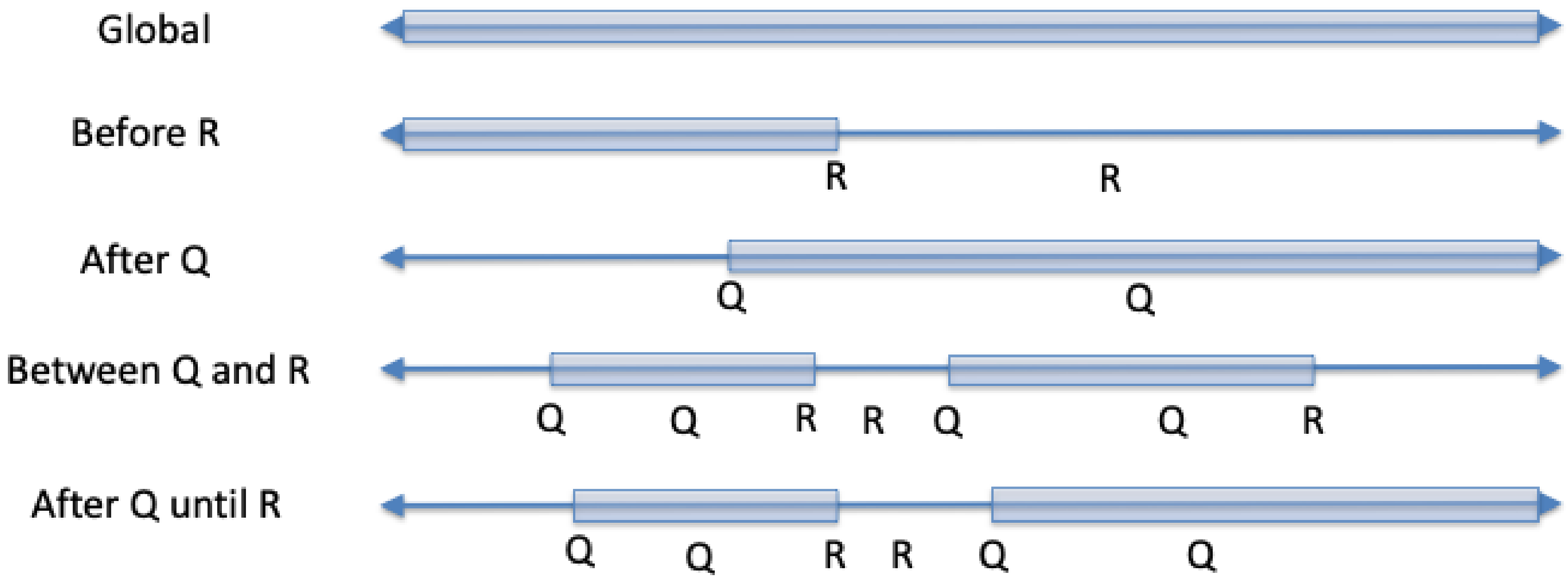
To describe cause-effect relationships between a pair of events/states. An occurrence of the first, the cause, must be followed by an occurrence of the second, the effect.

Also known as **Follows** and **Leads-to**.

S responds to P :

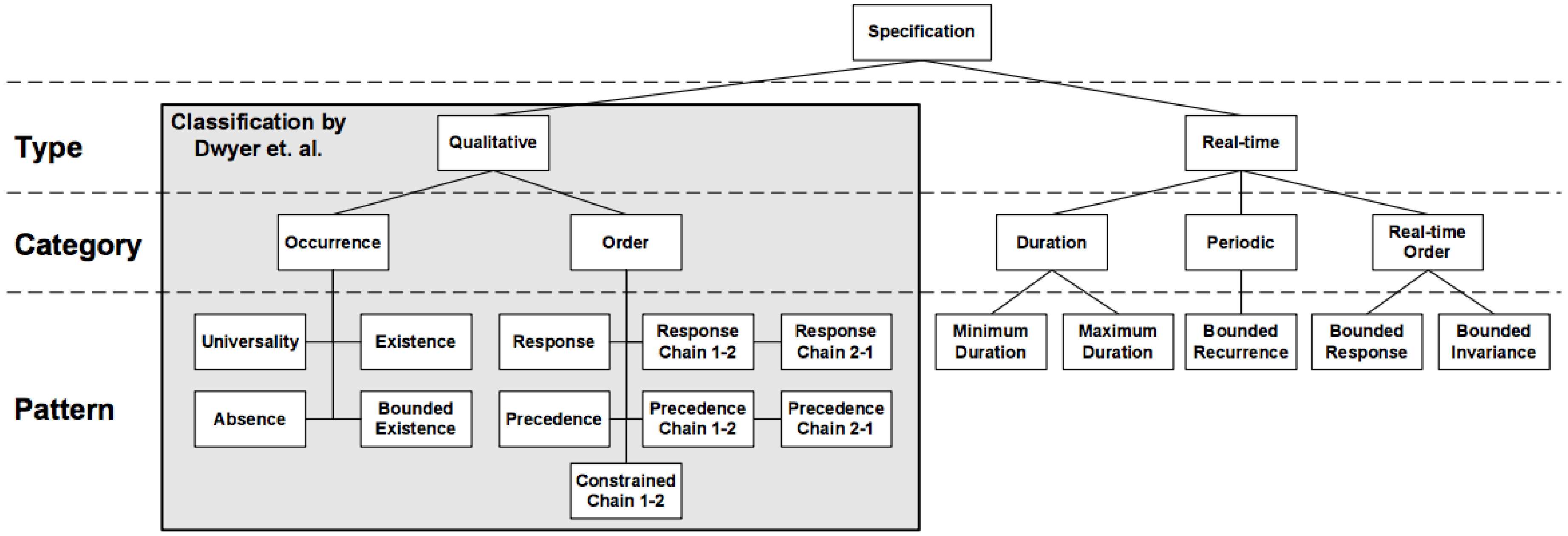
Globally	`[](P -> <>S)`
(*) Before R	`<>R -> (P -> (!R U (S & !R))) U R`
After Q	`[](Q -> [](P -> <>S))`
(*) Between Q and R	`[]((Q & !R & <>R) -> (P -> (!R U (S & !R))) U R)`
(*) After Q until R	`[](Q & !R -> ((P -> (!R U (S & !R))) W R)`





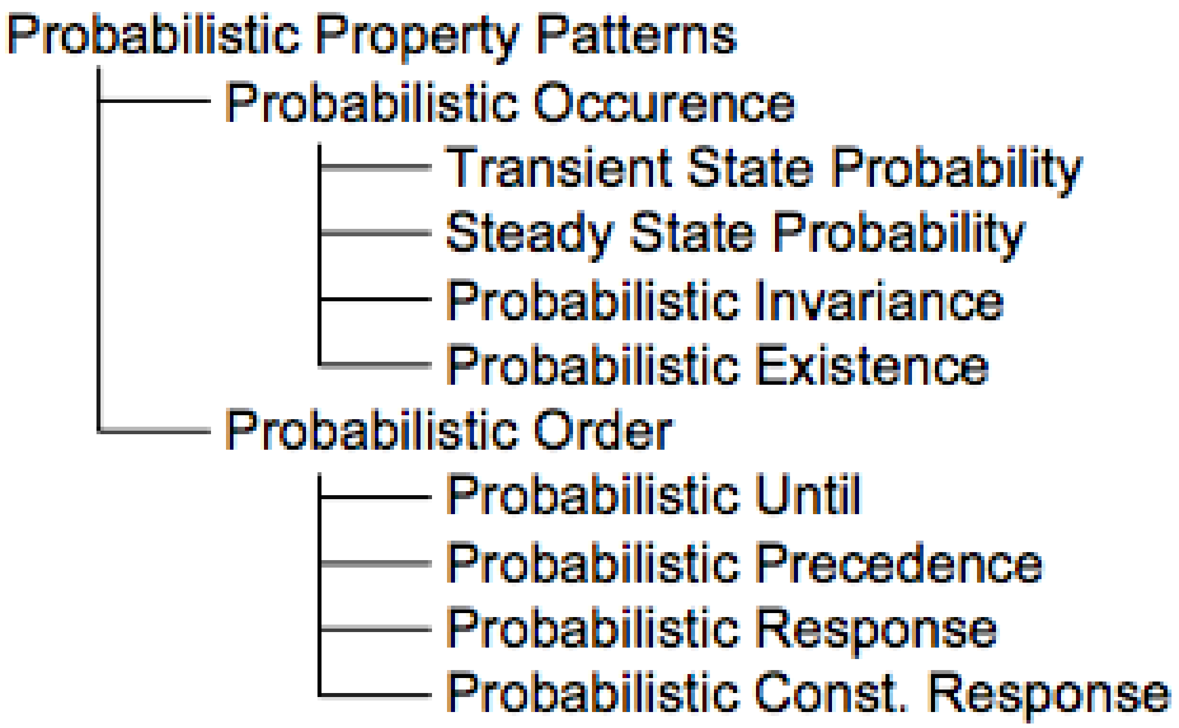
Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. 1999. **Patterns in property specifications for finite-state verification**. In *Proceedings of the 21st international conference on Software engineering* (ICSE '99). ACM, New York, NY, USA, 411-420.

Real-time specification patterns



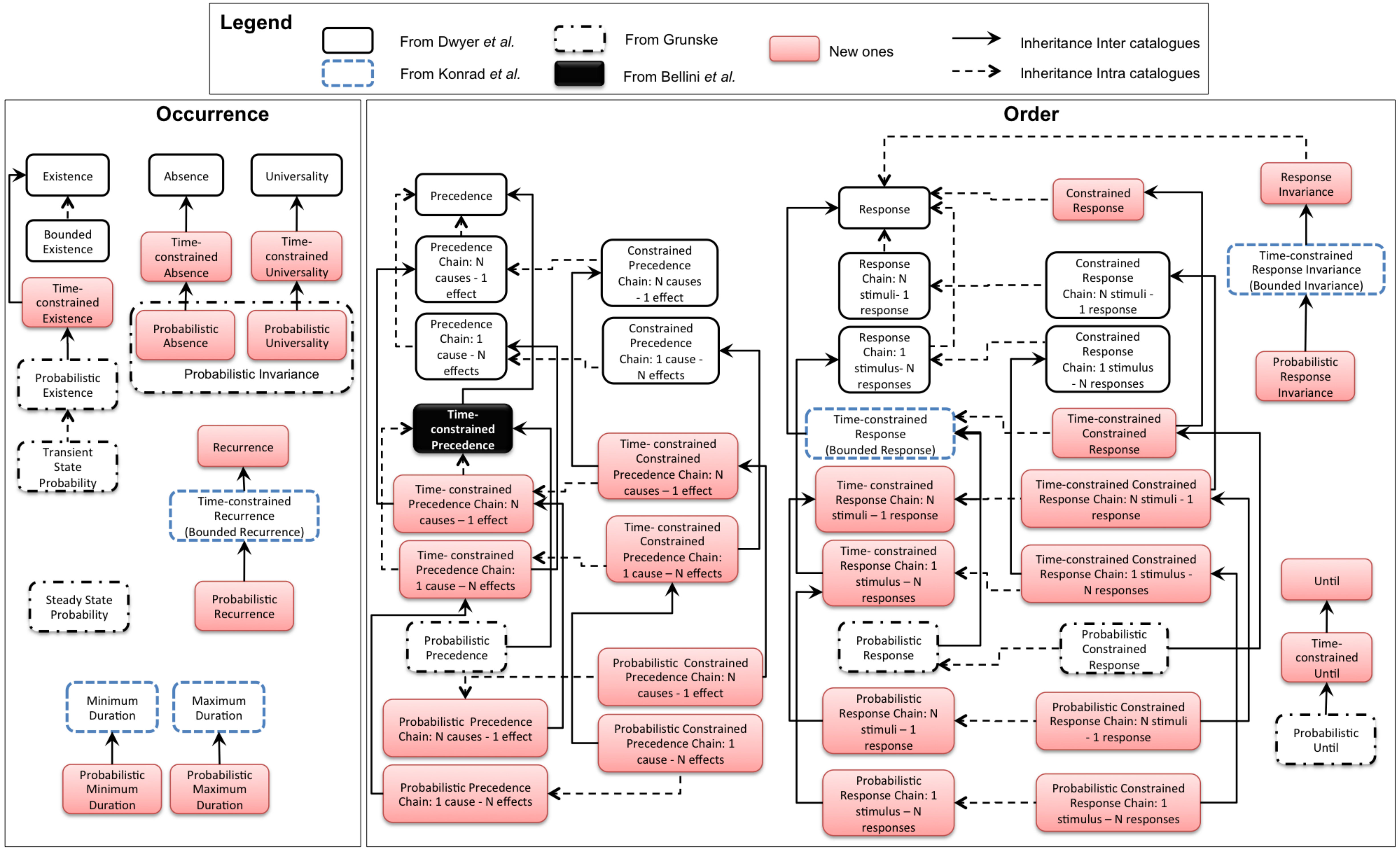
Sascha Konrad and Betty H. C. Cheng. 2005. **Real-time specification patterns**. In *Proceedings of the 27th international conference on Software engineering* (ICSE '05). ACM, New York, NY, USA, 372-381.

Probabilistic Property patterns



Lars Grunske. 2008. **Specification patterns for probabilistic quality properties**. In *Proceedings of the 30th international conference on Software engineering* (ICSE '08). ACM, New York, NY, USA, 31-40.

Unified catalogue of Property specification patterns



Integration of existing catalogues
+
40 newly identified or extended patterns

Marco Autili, Lars Grunske, Markus Lumpe, Patrizio Pelliccione, and Antony Tang (2015) **Aligning Qualitative, Real-Time, and Probabilistic Property Specification Patterns Using a Structured English Grammar**, IEEE Transactions on Software Engineering (TSE), , vol. 41, no. 7, pp. 620-638, 1 July 2015, doi: 10.1109/TSE.2015.2398877

Property Specification Patterns and Structured English grammar

`Precedence:`

Globally	::=	$\Box\,(\Diamond^{[\![\mathrm{trigger}(P)]\!]}\,P \rightarrow \Diamond^{[\![\mathrm{gap}(P)]\!]}\,S)$
Before $\{R\}$	::=	$\Diamond\,R \rightarrow (\Diamond^{[\![\mathrm{trigger}(P)]\!]}\,P \rightarrow (\Diamond^{[\![\mathrm{gap}(P)]\!]}\,S \vee \Diamond^{[\![\mathrm{elapsed}(P)]\!]}\,R))\,\mathcal{U}\,R$
After $\{Q\}$	::=	$\Box\,(Q \rightarrow \Box\,(\Diamond^{[\![\mathrm{trigger}(P)]\!]}\,P \rightarrow \Diamond^{[\![\mathrm{gap}(P)]\!]}\,S))$
Between $\{Q\}$ **and** $\{R\}$	::=	$\Box\,((Q \wedge \neg R \wedge \Diamond\,R) \rightarrow (\Diamond^{[\![\mathrm{trigger}(P)]\!]}\,P \rightarrow (\Diamond^{[\![\mathrm{gap}(P)]\!]}\,S \vee \Diamond^{[\![\mathrm{elapsed}(P)]\!]}\,R))\,\mathcal{U}\,R)$
After $\{Q\}$ **until** $\{R\}$	::=	$\Box\,((Q \wedge \neg R) \rightarrow (\Diamond^{[\![\mathrm{trigger}(P)]\!]}\,P \rightarrow (\Diamond^{[\![\mathrm{gap}(P)]\!]}\,S \vee \Diamond^{[\![\mathrm{elapsed}(P)]\!]}\,R))\,\mathcal{W}\,R)$

`PrecedenceChain`$_{1N}$`:`

Globally	::=	$\Box\,(\Diamond^{[\![\mathrm{trigger}(S)]\!]}\,(S\,[\![Ch(1)]\!]) \rightarrow \Diamond^{[\![\mathrm{maxgap}(S)]\!]}\,P)$
Before $\{R\}$	::=	$\Diamond\,R \rightarrow (\Diamond^{[\![\mathrm{trigger}(S)]\!]}\,(S\,[\![Ch(1)]\!])) \rightarrow (\Diamond^{[\![\mathrm{maxgap}(S)]\!]}\,P \vee \Diamond^{[\![\mathrm{gap}(N-1,S)]\!]}\,R)\,\mathcal{U}\,R)$
After $\{Q\}$	::=	$\Box\,(Q \rightarrow \Box\,(P \rightarrow (\Diamond^{[\![\mathrm{trigger}(S)]\!]}\,(S\,[\![Ch(1)]\!]))))$
Between $\{Q\}$ **and** $\{R\}$	::=	$\Box\,((Q \wedge \neg R \wedge \Diamond\,R) \rightarrow ((\Diamond^{[\![\mathrm{trigger}(S)]\!]}\,(S\,[\![Ch(1)]\!])) \rightarrow (\Diamond^{[\![\mathrm{maxgap}(S)]\!]}\,P \vee \Diamond^{[\![\mathrm{gap}(N-1,S)]\!]}\,R))\,\mathcal{U}\,R)$
After $\{Q\}$ **until** $\{R\}$	::=	$\Box\,((Q \wedge \neg R) \rightarrow ((\Diamond^{[\![\mathrm{trigger}(S)]\!]}\,(S\,[\![Ch(1)]\!])) \rightarrow (\Diamond^{[\![\mathrm{maxgap}(S)]\!]}\,P \vee \Diamond^{[\![\mathrm{gap}(N-1,S)]\!]}\,R))\,\mathcal{W}\,R)$
with $[\![Ch(i)]\!] = \wedge \bigcirc (\Diamond^{[\![\mathrm{utb}(T_i)]\!]}\,(T_i\,[\![Ch(i+1)]\!]))$		

`ConstrainedPrecedenceChain`$_{1N}$`:`

Globally	::=	$\Box\,([\![\mathrm{cnt}(S)]\!]\mathcal{U}^{[\![\mathrm{trigger}(S)]\!]}\,(S\,[\![$ …
Before $\{R\}$	::=	$\Diamond\,R \rightarrow ([\![\mathrm{cnt}(S)]\!]\mathcal{U}^{[\![\mathrm{trigger}(S)]\!]}$ …
After $\{Q\}$	::=	$\Box\,(Q \rightarrow \Box\,(P \rightarrow ([\![\mathrm{cnt}(S)]\!]\mathcal{U}$ …
Between $\{Q\}$ **and** $\{R\}$	::=	$\Box\,((Q \wedge \neg R \wedge \Diamond\,R) \rightarrow (([\![\mathrm{cnt}$ …
After $\{Q\}$ **until** $\{R\}$	::=	$\Box\,((Q \wedge \neg R) \rightarrow (([\![\mathrm{cnt}(S)]\!]\mathcal{U}$ …
with $[\![Ch(i)]\!] = \wedge [\![\mathrm{cnt}(T_i)]\!] \wedge \bigcirc([\![\mathrm{cnt}(T_i)]\!]\,\mathcal{U}^{[\![\mathrm{utb}(T_i$ …		

`PrecedenceChain`$_{N1}$`:`

Globally	::=	$\Box\,(\Diamond^{[\![\mathrm{trigger}(P)]\!]}\,P \rightarrow (\Diamond^{[\![\mathrm{gap}(P}$ …
Before $\{R\}$	::=	$\Diamond\,R \rightarrow ((\Diamond^{[\![\mathrm{trigger}(P)]\!]}\,P \rightarrow ($ …
After $\{Q\}$	::=	$\Box\,(Q \rightarrow \Box\,(\Diamond^{[\![\mathrm{trigger}(P)]\!]}\,P \rightarrow$ …
Between $\{Q\}$ **and** $\{R\}$	::=	$\Box\,((Q \wedge \neg R \wedge \Diamond\,R) \rightarrow ((\Diamond^{[\![\mathrm{tr}}$ …
After $\{Q\}$ **until** $\{R\}$	::=	$\Box\,((Q \wedge \neg R) \rightarrow ((\Diamond^{[\![\mathrm{trigger}(P)}$ …
with $[\![Ch(i)]\!] = \wedge \bigcirc (\Diamond^{[\![\mathrm{gap}(P,i)]\!]}\,(T_i\,[\![Ch(i+1)]\!]))$		

`Property`	::=	`Scope, Pattern.`
`Scope`	::=	**Globally** | **Before** $\{R\}$ | **After** $\{Q\}$ | **Between** $\{Q\}$ **and** $\{R\}$ | **After** $\{Q\}$ **until** $\{R\}$
`Pattern`	::=	`Occurrence` | `Order`
`Occurrence`	::=	`Universality` | `Absence` | `Existence` | `BoundedExistence` | `TransientState` | `SteadyState` | `MinimumDuration` | `MaximumDuration` | `Recurrence`
`Universality`	::=	**it is always the case that** $\{P\}$ [**holds**] [`Time` (P)] [`Probability`]
`Absence`	::=	**it is never the case that** $\{P\}$ [**holds**] [`Time` (P)] [`Probability`]
`Existence`	::=	$\{P\}$ [**holds**] **eventually** [`Time` (P)] [`Probability`]
`BoundedExistence`	::=	$\{P\}$ [**holds**] **at most** n **times** [`Time` (P)] [`Probability`]
`TransientState`	::=	$\{P\}$ [**holds**] **after** t_u^P `TimeUnits` [`Probability`]
`SteadyState`	::=	$\{P\}$ [**holds**] **in the long run** [`Probability`]
`MinimumDuration`	::=	**once** $\{P\}$ [**becomes satisfied**] **it remains so for at least** t_u^P `TimeUnits` [`Probability`]
`MaximumDuration`	::=	**once** $\{P\}$ [**becomes satisfied**] **it remains so for less than** t_u^P `TimeUnits` [`Probability`]
`Recurrence`	::=	$\{P\}$ [**holds**] **repeatedly** [**every** t_u^P `TimeUnits`] [`Probability`]
`Order`	::=	`Precedence` | `PrecedenceChain`$_{1N}$ | `PrecedenceChain`$_{N1}$ | `Until` | `Response` | `ResponseChain`$_{1N}$ | `ResponseChain`$_{N1}$ | `ResponseInvariance`
`Precedence`	::=	**if** $\{P\}$ [**holds**] **then it must have been the case that** $\{S\}$ [**has occurred**] [`Interval` (P)] **before** $\{P\}$ [**holds**] [`Probability`]
`PrecedenceChain`$_{1N}$	::=	**if** $\{S\}$ [**has occurred**] **and afterwards** $(\{T_i\}$ [`UpperTimeBound` (T_i)] [`Constraint` (T_i)]$)^{(1 \le i \le N-1; \text{“,”})}$ [**hold**] **then it must have been the case that** $\{P\}$ [**has occurred**] [`Interval` (S)] **before** $\{S\}$ [**holds**] [`Constraint` (S)] [`Probability`]

Marco Autili, Lars Grunske, Markus Lumpe, Patrizio Pelliccione, and Antony Tang (2015) **Aligning Qualitative, Real-Time, and Probabilistic Property Specification Patterns Using a Structured English Grammar**, IEEE Transactions on Software Engineering (TSE), , vol. 41, no. 7, pp. 620-638, 1 July 2015, doi: 10.1109/TSE.2015.2398877

Property Specification Patterns and Structured English grammar

Precedence:

Globally	::=	$\Box(\Diamond^{[\mathrm{trigger}(P)]} P \rightarrow \Diamond^{[\mathrm{gap}(P)]} S)$
Before $\{R\}$	::=	$\Diamond R \rightarrow (\Diamond^{[\mathrm{trigger}(P)]} P \rightarrow (\Diamond^{[\mathrm{gap}(P)]} S \vee \Diamond^{[\mathrm{elapsed}(P)]} R))\ \mathcal{U}\ R$
After $\{Q\}$	::=	$\Box(Q \rightarrow \Box(\Diamond^{[\mathrm{trigger}(P)]} P \rightarrow \Diamond^{[\mathrm{gap}(P)]} S))$
Between $\{Q\}$ and $\{R\}$	::=	$\Box((Q \wedge \neg R \wedge \Diamond R) \rightarrow (\Diamond^{[\mathrm{trigger}(P)]} P \rightarrow (\Diamond^{[\mathrm{gap}(P)]} S \vee \Diamond^{[\mathrm{elapsed}(P)]} R))\ \mathcal{U}\ R)$
After $\{Q\}$ until $\{R\}$	::=	$\Box((Q \wedge \neg R) \rightarrow (\Diamond^{[\mathrm{trigger}(P)]} P \rightarrow (\Diamond^{[\mathrm{gap}(P)]} S \vee \Diamond^{[\mathrm{elapsed}(P)]} R))\ \mathcal{W}\ R)$

PrecedenceChain$_{1N}$:

Globally	::=	$\Box(\Diamond^{[\mathrm{trigger}(S)]} (S\,[\![Ch(1)]\!]) \rightarrow \Diamond^{[\mathrm{maxgap}(S)]} P)$
Before $\{R\}$	::=	$\Diamond R \rightarrow (\Diamond^{[\mathrm{trigger}(S)]} (S\,[\![Ch(1)]\!])) \rightarrow (\Diamond^{[\mathrm{maxgap}(S)]} P \vee \Diamond^{[\mathrm{gap}(N-1,S)]} R)\ \mathcal{U}\ R)$
After $\{Q\}$	::=	$\Box(Q \rightarrow \Box(P \rightarrow (\Diamond^{[\mathrm{trigger}(S)]} (S\,[\![Ch(1)]\!]))))$
Between $\{Q\}$ and $\{R\}$	::=	$\Box((Q \wedge \neg R \wedge \Diamond R) \rightarrow ((\Diamond^{[\mathrm{trigger}(S)]} (S\,[\![Ch(1)]\!])) \rightarrow (\Diamond^{[\mathrm{maxgap}(S)]} P \vee \Diamond^{[\mathrm{gap}(N-1,S)]} R))\ \mathcal{U}\ R)$
After $\{Q\}$ until $\{R\}$	::=	$\Box((Q \wedge \neg R) \rightarrow ((\Diamond^{[\mathrm{trigger}(S)]} (S\,[\![Ch(1)]\!])) \rightarrow (\Diamond^{[\mathrm{maxgap}(S)]} P \vee \Diamond^{[\mathrm{gap}(N-1,S)]} R))\ \mathcal{W}\ R)$
with $[\![Ch(i)]\!] = \wedge \bigcirc (\Diamond^{[\mathrm{utb}(T_i)]} (T_i\,[\![Ch(i+1)]\!]))$		

ConstrainedPrecedenceChain$_{1N}$:

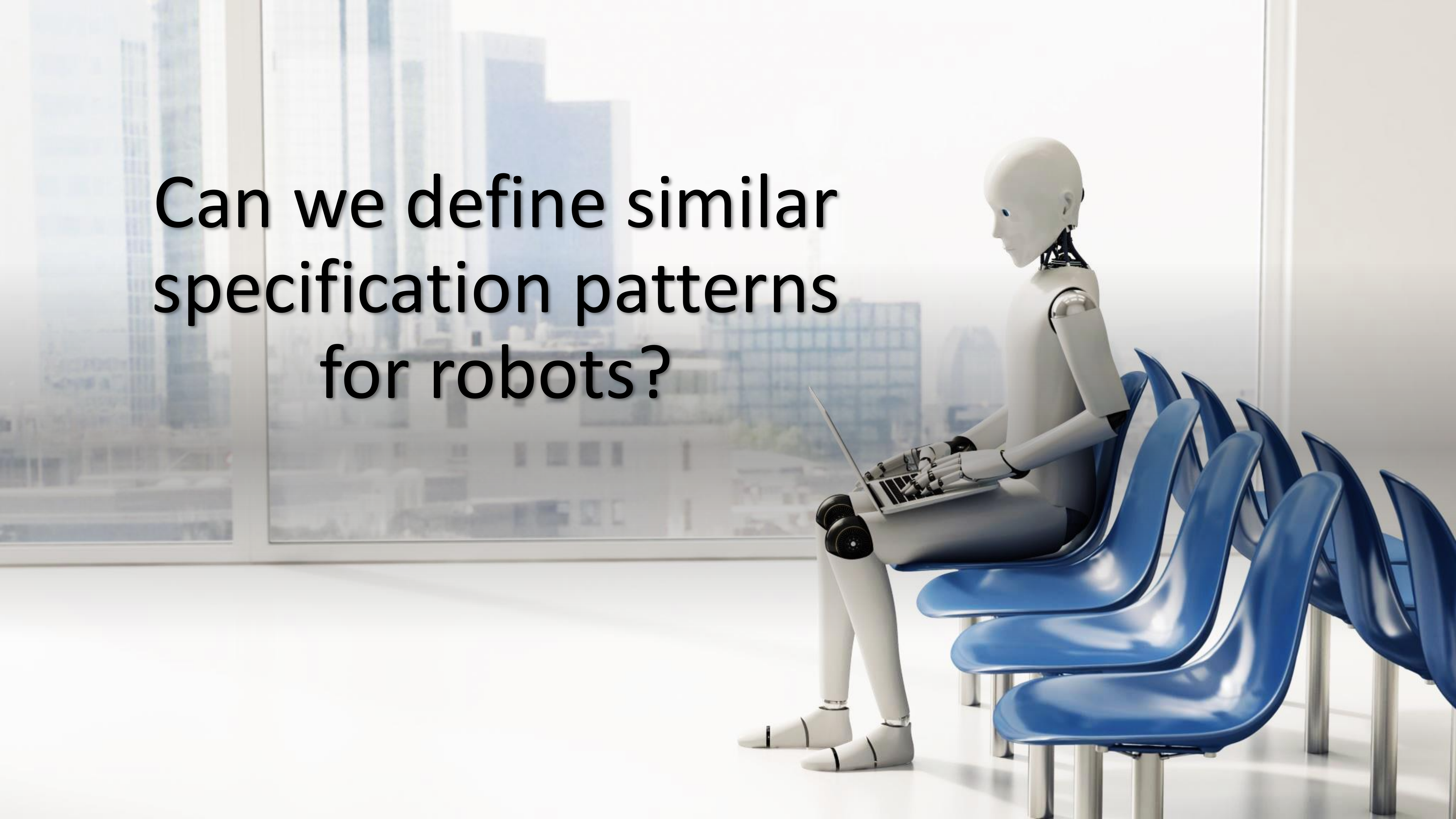
Globally	::=	$\Box([\mathrm{cnt}(S)]\mathcal{U}^{[\mathrm{trigger}(S)]} (S\,[\![Ch(1)]\!]) \rightarrow \Diamond^{[\mathrm{maxgap}(S)]} P))$
Before $\{R\}$	::=	$\Diamond R \rightarrow ([\mathrm{cnt}(S)]\mathcal{U}^{[\mathrm{trigger}}$ …
After $\{Q\}$	::=	$\Box(Q \rightarrow \Box(P \rightarrow ([\mathrm{cnt}(S$ …
Between $\{Q\}$ and $\{R\}$	::=	$\Box((Q \wedge \neg R \wedge \Diamond R) \rightarrow (([$ …
After $\{Q\}$ until $\{R\}$	::=	$\Box((Q \wedge \neg R) \rightarrow (([\mathrm{cnt}(S$ …
with $[\![Ch(i)]\!] = \wedge [\mathrm{cnt}(T_i)] \wedge \bigcirc([\mathrm{cnt}(T_i)]\ \mathcal{U}^{[\mathrm{ut}}$ …		

PrecedenceChain$_{N1}$:

Globally	::=	$\Box(\Diamond^{[\mathrm{trigger}(P)]} P \rightarrow (\Diamond^{[\mathrm{g}}$ …
Before $\{R\}$	::=	$\Diamond R \rightarrow ((\Diamond^{[\mathrm{trigger}(P)]} P$ …
After $\{Q\}$	::=	$\Box(Q \rightarrow \Box(\Diamond^{[\mathrm{trigger}(P)]}$ …
Between $\{Q\}$ and $\{R\}$	::=	$\Box((Q \wedge \neg R \wedge \Diamond R) \rightarrow (($ …
After $\{Q\}$ until $\{R\}$	::=	$\Box((Q \wedge \neg R) \rightarrow ((\Diamond^{[\mathrm{trigge}}$ …
with $[\![Ch(i)]\!] = \wedge \bigcirc (\Diamond^{[\mathrm{gap}(P,i)]} (T_i\,[\![Ch(i+1)]\!]$ …		

Property	::=	Scope, Pattern.
Scope	::=	**Globally** | **Before** $\{R\}$ | **After** $\{Q\}$ | **Between** $\{Q\}$ **and** $\{R\}$ | **After** $\{Q\}$ **until** $\{R\}$
Pattern	::=	Occurrence | Order
Occurrence	::=	Universality | Absence | Existence | BoundedExistence | TransientState | SteadyState | MinimumDuration | MaximumDuration | Recurrence
Universality	::=	**it is always the case that** $\{P\}$ [holds] [Time (P)] [Probability]
Absence	::=	**it is never the case that** $\{P\}$ [holds] [Time (P)] [Probability]
Existence	::=	$\{P\}$ [holds] **eventually** [Time (P)] [Probability]
BoundedExistence	::=	$\{P\}$ [holds] **at most** n **times** [Time (P)] [Probability]
TransientState	::=	$\{P\}$ [holds] **after** t_u^P TimeUnits [Probability]
SteadyState	::=	$\{P\}$ [holds] **in the long run** [Probability]
MinimumDuration	::=	**once** $\{P\}$ [**becomes satisfied**] **it remains so for at least** t_u^P TimeUnits [Probability]
MaximumDuration	::=	**once** $\{P\}$ [**becomes satisfied**] **it remains so for less than** t_u^P TimeUnits [Probability]
Recurrence	::=	$\{P\}$ [holds] **repeatedly** [**every** t_u^P TimeUnits] [Probability]
Order	::=	Precedence | PrecedenceChain$_{1N}$ | PrecedenceChain$_{N1}$ | Until| Response| ResponseChain$_{1N}$| ResponseChain$_{N1}$| ResponseInvariance
Precedence	::=	**if** $\{P\}$ [holds] **then it must have been the case that** $\{S\}$ [has occurred] [Interval (P)] **before** $\{P\}$ [holds] [Probability]
PrecedenceChain$_{1N}$	::=	**if** $\{S\}$ [**has occurred**] **and afterwards** $(\{T_i\}$ [UpperTimeBound (T_i)] [Constraint (T_i)]$)^{(1\leq i\leq N-1;\text{“,”})}$ [hold] **then it must have been the case that** $\{P\}$ [has occurred] [Interval (S)] **before** $\{S\}$ [holds] [Constraint (S)] [Probability]

Marco Autili, Lars Grunske, Markus Lumpe, Patrizio Pelliccione, and Antony Tang (2015) **Aligning Qualitative, Real-Time, and Probabilistic Property Specification Patterns Using a Structured English Grammar**, IEEE Transactions on Software Engineering (TSE), , vol. 41, no. 7, pp. 620-638, 1 July 2015, doi: 10.1109/TSE.2015.2398877


Can we define similar
specification patterns
for robots?

Let’s first discuss another important aspect

Simplicity but keeping rigorousness

In this context, Ambiguity is evil!

Intuitive and Simple but also Rigorous

"A robot r shall visit the two locations l1 and l2 in this order"

Intuitive and Simple but also Rigorous

"A robot r shall visit the two locations l1 and l2 in this order"

l1 and then l2

Intuitive and Simple but also Rigorous

Ambiguity

"A robot r shall visit the two locations l1 and l2 in this order"

l1 and then l2

Is it possible to visit l2 before l1 and then to visit l2?

Intuitive and Simple but also Rigorous

"A robot r shall visit the two locations l1 and l2 in this order"

l1 and then l2

Is it possible to visit l2 before l1 and then to visit l2?

Φ1=<>((r in l1) && <>(r in l2)) VS. ϕ2 = ϕ1 && ((!r in l2)U(r in l1))

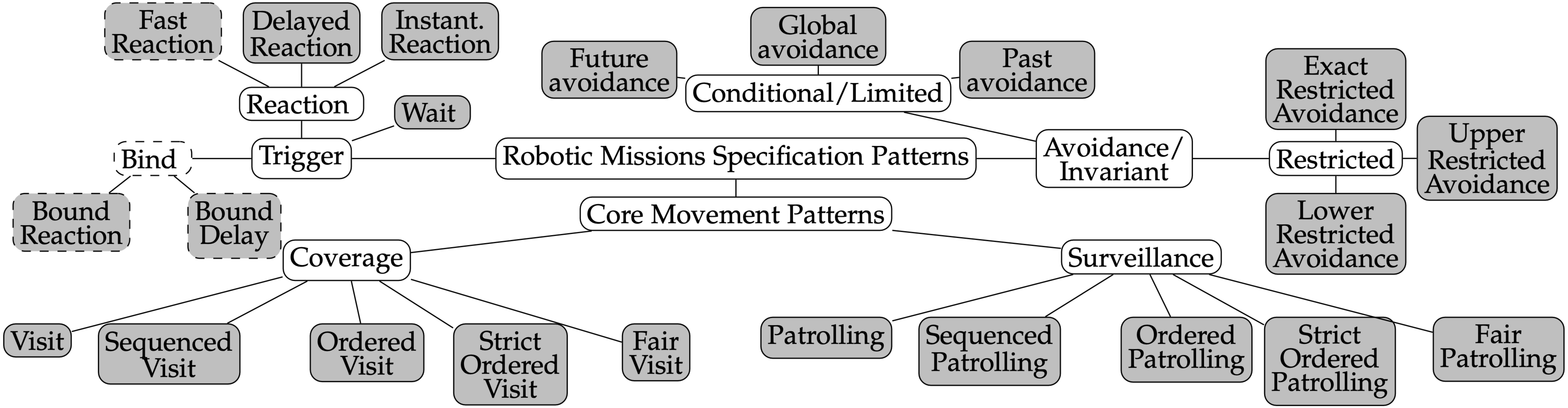
Intuitive and Simple but also Rigorous

If we allow an ambiguous specification,
which behavior will have the robot,
and who will decide it?

This is why ambiguity is evil!

Mission specification patterns for robots

Specification Patterns



Claudio Menghi, Christos Tsigkanos, Patrizio Pelliccione, Carlo Ghezzi, and Thorsten Berger, Specification Patterns for Robotic Missions, Transactions on Software Engineering (TSE), 2019

http://roboticpatterns.com

Specification Patterns

	Description	Example	Formula ($l_1, l_2, \ldots$ are location propositions)
Visit	Visit a set of locations in an unspecified order.	Locations l_1, l_2, and l_3 must be visited. $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow (l_\#)^\omega$ is an example trace that satisfies the mission requirement.	$\bigwedge_{i=1}^{n} \mathcal{F}(l_i)$
Sequenced Visit	Visit a set of locations in sequence, one after the other.	Locations l_1, l_2, l_3 must be covered following this sequence. The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow (l_{\#\backslash 3})^\omega$ violates the mission since l_3 does not follow l_2. The trace $l_1 \rightarrow l_3 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_\#)^\omega$ satisfies the mission requirement.	$\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \ldots \mathcal{F}(l_n)))$
Ordered Visit	The sequenced visit pattern does not forbid to visit a successor location before its predecessor, but only that after the predecessor is visited the successor is also visited. Ordered visit forbids a successor to be visited before its predecessor.	Locations l_1, l_2, l_3 must be covered following this order. The trace $l_1 \rightarrow l_3 \rightarrow l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow (l_\#)^\omega$ does not satisfy the mission requirement since l_3 preceeds l_2. The trace $l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_\#)^\omega$ satisfies the mission requirement.	$\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \ldots \mathcal{F}(l_n)))$ $\bigwedge_{i=1}^{n-1} (\neg l_{i+1}) \,\mathcal{U}\, l_i$
Strict Ordered Visit	The ordered visit pattern does not avoid a predecessor location to be visited multiple times before its successor. Strict ordered visit forbids this behavior.	Locations l_1, l_2, l_3 must be covered following the strict order l_1, l_2, l_3. The trace $l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_\#)^\omega$ does not satisfy the mission requirement since l_1 occurs twice before l_2. The trace $l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_\#)^\omega$ satisfies the mission requirement.	$\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \ldots \mathcal{F}(l_n)))$ $\bigwedge_{i=1}^{n-1} (\neg l_{i+1}) \,\mathcal{U}\, l_i$ $\bigwedge_{i=1}^{n-1} (\neg l_i) U (l_i \wedge \mathcal{X}(\neg l_i \,\mathcal{U}(l_{i+1})))$
Fair Visit	The difference among the number of times locations within a set are visited is at most one.	Locations l_1, l_2, l_3 must be covered in a fair way. The trace $l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow (l_{\#\backslash\{1,2,3\}})^\omega$ does not perform a fair visit since it visits l_1 three times while l_2 and l_3 are visited once. The trace $l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow l_2 \rightarrow l_4 \rightarrow (l_{\#\backslash\{1,2,3\}})^\omega$ performs a fair visit since it visits locations l_1, l_2, and l_3 twice.	$\bigwedge_{i=1}^{n} \mathcal{F}(l_i)$ $\bigwedge_{i=1}^{n} \mathcal{G}(l_i \rightarrow \mathcal{X}((\neg l_i) \,\mathcal{W}\, l_{(i+1)\%n}))$

Claudio Menghi, Christos Tsigkanos, Patrizio Pelliccione, Carlo Ghezzi, and Thorsten Berger, Specification Patterns for Robotic Missions, Transactions on Software Engineering (TSE), 2019

http://roboticpatterns.com

An example of pattern

Name: Strict Ordered Patrolling

Intent: A robot must patrol a set of locations following a strict sequence ordering. Such locations can be, e.g., areas in a building to be surveyed.

Template: The following formula encodes the mission in LTL for n locations and a robot r (% is the modulo arithmetic operator):

$$\bigwedge_{i=1}^{n} \mathcal{G}(\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \ldots \mathcal{F}(l_n)))) \bigwedge_{i=1}^{n-1} ((\neg l_{i+1})\ \mathcal{U}\ l_i) \bigwedge_{i=1}^{n} \mathcal{G}(l_{(i+1)\%n} \rightarrow \mathcal{X}((\neg l_{(i+1)\%n})\ \mathcal{U}\ l_i))$$

Example with two locations.

$$\mathcal{G}(\mathcal{F}(l_1 \wedge \mathcal{F}(l_2))) \wedge ((\neg l_2)\ \mathcal{U}\ l_1) \wedge \mathcal{G}(l_2 \rightarrow \mathcal{X}((\neg l_2)\ \mathcal{U}\ l_1)) \wedge \mathcal{G}(l_1 \rightarrow \mathcal{X}((\neg l_1)\ \mathcal{U}\ l_2))$$

where l_1 and l_2 are expressions that indicate that a robot r is in locations l_1 and l_2, respectively.

Variations: A developer may want to allow traces in which sequences of *consecutive* l_1 (l_2) are allowed, that is strict ordering is applied on sequences of non consecutive l_1 (l_2). In this case, traces in the form $l_1 \rightarrow (\rightarrow l_1 \rightarrow l_1 \rightarrow l_3 \rightarrow l_2)^\omega$ are admitted, while traces in the form $l_1 \rightarrow (\rightarrow l_1 \rightarrow l_3 \rightarrow l_1 \rightarrow l_2)^\omega$ are not admitted. This variation can be encoded using the following specification:

$$\mathcal{G}(\mathcal{F}(l_1 \wedge \mathcal{F}(l_2))) \wedge ((\neg l_2)\ \mathcal{U}\ l_1) \wedge \mathcal{G}((l_2 \wedge \mathcal{X}(\neg l_2)) \rightarrow \mathcal{X}((\neg l_2)\ \mathcal{U}\ l_1)) \wedge \mathcal{G}((l_1 \wedge \mathcal{X}(\neg l_1)) \rightarrow \mathcal{X}((\neg l_1)\ \mathcal{U}\ l_2))$$

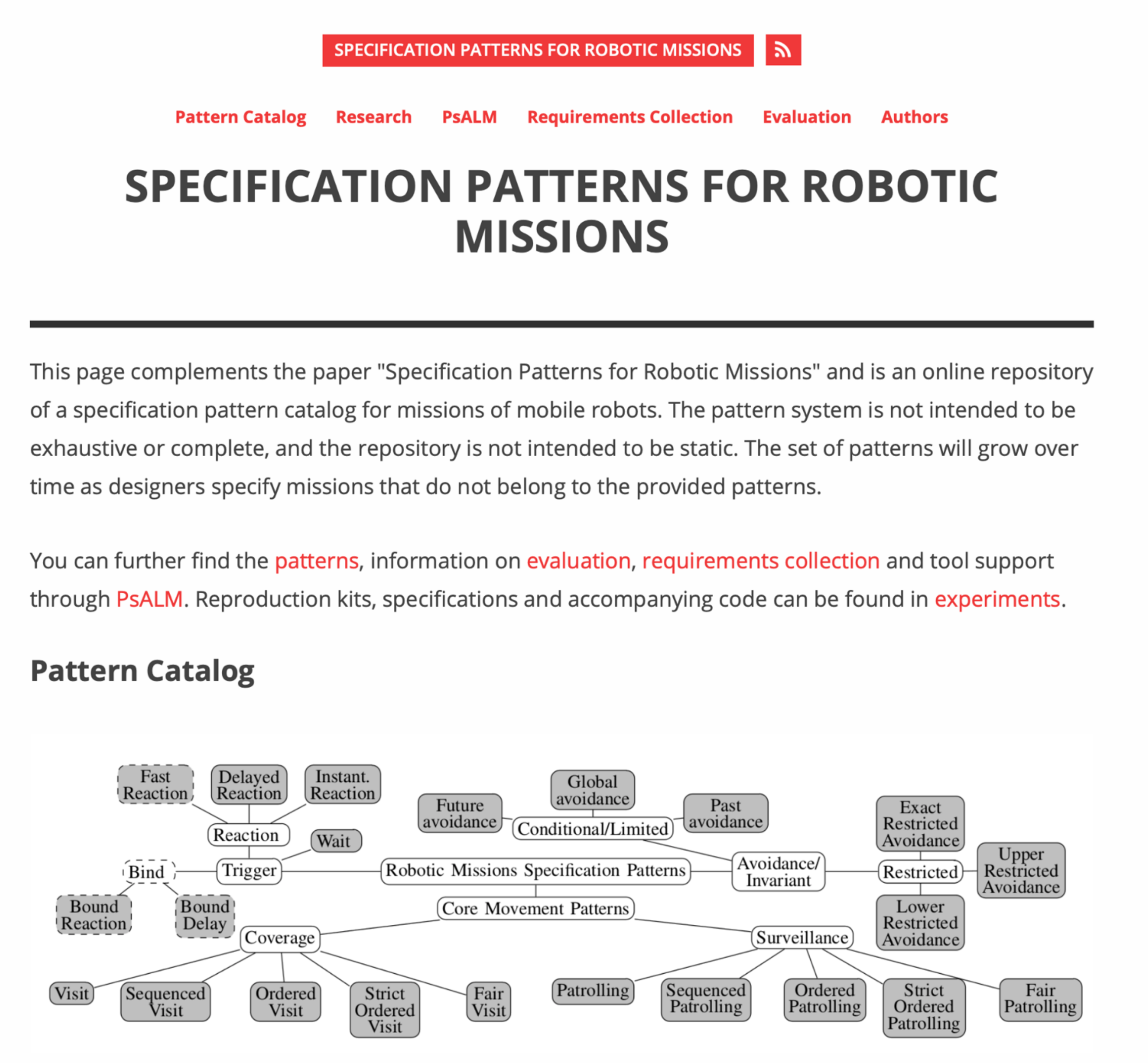
This specification allows for sequences of consecutive l_1 (l_2) since the left side of the implication $l_1 \wedge \mathcal{X}(\neg l_1)$ ($l_2 \wedge \mathcal{X}(\neg l_2)$) is only triggered when l_1 (l_2) is exited.

Examples and Known Uses: A common usage example of the Strict Ordered Patrolling pattern is a scenario where a robot is performing surveillance in a building during night hours. Strict Sequence Patrolling and Avoidance often go together. Avoidance patterns are used to force robots to avoid obstacles as they guard a location. Triggers can also be used in combination with the Strict Sequence Patrolling pattern to specify conditions upon which Patrolling should start or stop.

Relationships: The Strict Ordered Patrolling pattern is a specialisation of the Ordered Patrolling pattern, forcing the strict ordering.

Occurrences: Smith et. al. [74] proposed a mission specification forcing a robot to not visit a location twice in a row before a target location is reached.

Further info about specification patterns


SPECIFICATION PATTERNS FOR ROBOTIC MISSIONS

Pattern Catalog | Research | PsALM | Requirements Collection | Evaluation | Authors

SPECIFICATION PATTERNS FOR ROBOTIC MISSIONS

This page complements the paper "Specification Patterns for Robotic Missions" and is an online repository of a specification pattern catalog for missions of mobile robots. The pattern system is not intended to be exhaustive or complete, and the repository is not intended to be static. The set of patterns will grow over time as designers specify missions that do not belong to the provided patterns.

You can further find the patterns, information on evaluation, requirements collection and tool support through PsALM. Reproduction kits, specifications and accompanying code can be found in experiments.

Pattern Catalog

http://roboticpatterns.com

Claudio Menghi, Christos Tsigkanos, Patrizio Pelliccione, Carlo Ghezzi, and Thorsten Berger, Specification Patterns for Robotic Missions, **Transactions on Software Engineering (TSE)**, 2019

Journal First-track at ICSE 2020

Claudio Menghi, Christos Tsigkanos, Thorsten Berger, and Patrizio Pelliccione, PsALM: Specification of Dependable Robotic Missions. IEEE/ACM **ICSE Demo**, 2019.

https://github.com/claudiomenghi/PsAlM

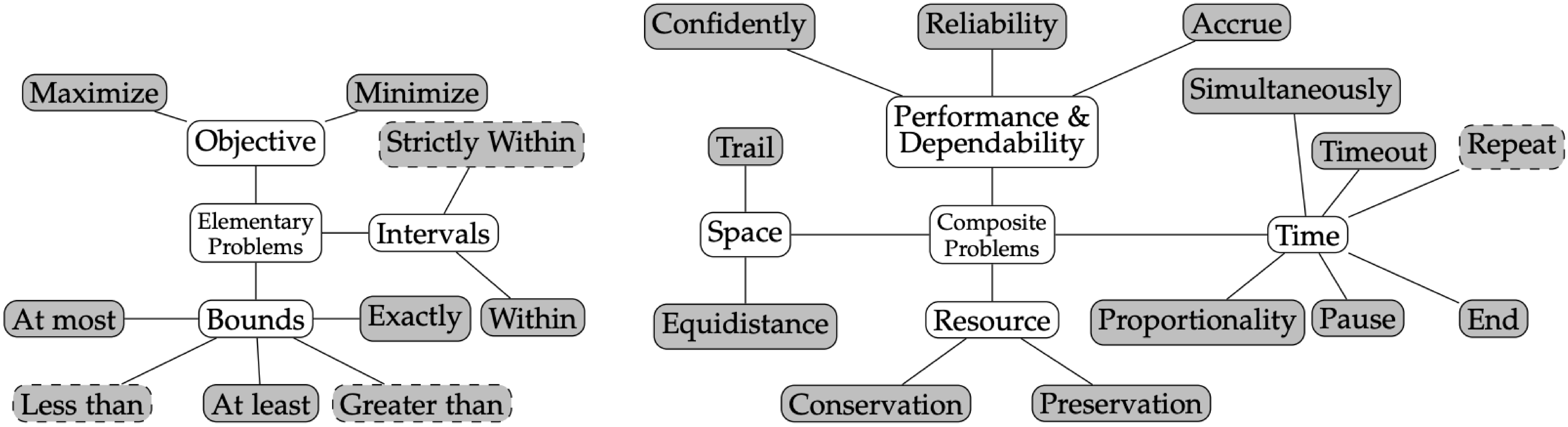
Are these patterns enough? Example of mission requirement

"After closure, the robots shall clean the electronics store. After cleaning, they shall visit a set of predefined store locations, each at least once, to record the items present on shelves after closure. The robots must minimize the time required to perform this activity. The robots should also patrol the store for security purposes, following any intruder while raising an alarm. The robots should interleave cleaning and security patrolling so that intruders do not remain undetected while the robots are cleaning continually for long periods of time. The robots should monitor their battery, optimize its usage, and recharge when needed. They should avoid recharging simultaneously and leaving the store unmonitored."

Support for quantitative aspects

Users and operators of robotic systems often require behaviors that ensure quantitative constraints such as **upper bounds** on the **time** a robot takes to perform an action, the **energy consumption** to complete that action, or the **probability of failing** to achieve a mission goal.

We extended previous patterns to support the specification of quantitative properties



(a) Elementary mission specification problems.

(b) Composite mission specification problems.

Claudio Menghi, Christos Tsigkanos, Mehrnoosh Askarpour , Patrizio Pelliccione, Gricel Vazquez , Radu Calinescu, and Sergio García "Mission Specification Patterns for Mobile Robots: Providing Support for Quantitative Properties," in **IEEE Transactions on Software Engineering (TSE)**, doi: 10.1109/TSE.2022.3230059.

Domain Specific Language (DSL) including previous patterns

Previous patterns: Pattern
New ones: Elementary Patterns, Composite Patterns

Mission	miss	::= miss **and** miss | miss **or** miss | **not** miss | rob **shall** pat | e_qpat | c_qpat
Pattern	pat	::= **visit** (**in sequence** | **in order** | **in strict order** | **fairly**)? locs | **patrol** (**in sequence** | **in order** | **in strict order** | **fairly**)? locs | **visit** (**more than** | **less than** | **exactly**) n **times** loc | **avoid** (loc **until** cond | loc | loc **after** cond) | **react** (**instantly** | **with a delay** | **promptly**) **to** cond **by** (**exec** act | pat | **reach** loc) | **counteract** (**instantly** | **with a delay**) **when reach** loc **by** cond **wait in location** loc **until** cond
Elementary Patterns	e_qpat	::= **maximize** m miss | **minimize** m miss | m **at most** v miss | m **less than** v miss | m **at least** v miss | m **greater than** v miss | m **exactly** v miss | m **within** v_1 **and** v_2 miss | m **strictly within** v_1 **and** v_2 miss
Composite Patterns	c_qpat	::= **conserve** m **while** miss | **preserve** m **within** [v_1,v_2] **while** miss | **pause** v miss | **timeout** v miss | **repeat** miss **every** v | **end** miss **exactly at** v | **time of** $miss_1$ **proportional to** $miss_2$ **by factor** v | **execute** rob **actions** act_1,act_2,… act_n | rob **accrue** m **while** miss | **achieve** miss **with reliability** m (**greater** | **less**) **than** v | **achieve** miss **with confidence** m (**greater** | **less**) **than** v | rob miss **equidistance** rob_1 rob_2 | rob **trail** o **with distance** v
Condition	cond	::= condition **is true** | act **is ended** | rob **in** loc
Locations	locs	::= {loc (, loc)*}

* miss, $miss_1$, $miss_2$ are missions; v, v_1, v_2 are values; rob is a robot, o is an object, m is the name of the quantitative measure.

Claudio Menghi, Christos Tsigkanos, Mehrnoosh Askarpour , Patrizio Pelliccione, Gricel Vazquez , Radu Calinescu, and Sergio García "Mission Specification Patterns for Mobile Robots: Providing Support for Quantitative Properties," in **IEEE Transactions on Software Engineering (TSE)**, doi: 10.1109/TSE.2022.3230059.

New “quantitative” patterns

Problem	Description	DSL
Maximize	Maximize m while performing the mission miss.	**maximize** m miss
Minimize	Minimize m while performing the mission miss.	**minimize** m miss
At most	Keep m lower than or equal to v while performing miss.	m **at most** v miss
Less than	Keep m strictly lower than v while performing miss.	m **less than** v miss
At least	Keep m greater than or equal to v while performing miss.	m **at least** v miss
Greater than	Keep m strictly greater than v while performing miss.	m **greater than** v miss
Exactly	Keep m exactly v while performing miss.	m **exactly** v miss
Within	Keep m within the (closed) interval $[v_1, v_2]$ while performing miss.	m **within** v_1 **and** v_2 miss
Strictly Within	Keep m within the (open) interval (v_1, v_2) while performing miss.	m **strictly within** v_1 **and** v_2 miss
Conservation	Minimize the value of m performing miss.	**conserve** m **while** miss
Preservation	Keep the value of m within interval $[b_l, b_u]$ while performing miss.	**preserve** m **within** $[v_1, v_2]$ **while** miss
Pause	Pause the mission miss for v time instants. Then, resume it.	**pause** v miss
Timeout-deadline	Execute miss. Stop the the execution when the timeout v is reached.	**timeout** v miss
Repeat	Repeat the mission miss every v time units.	**repeat** miss **every** v
End	Terminate mission miss exactly at time v.	**end** miss **exactly_at** v
Proportionality	Keep the time to perform $miss_1$ and $miss_2$ proportional by a factor v.	**time of** $miss_1$ **proportional to** [...]
Simultaneously	Execute the actions $act_1, act_2, \ldots, act_n$ simultaneously.	**execute** rob **actions** $act_1, act_2, \ldots act_n$
Accrue	Maximize the performance m while performing miss.	rob **accrue** m **while** miss
Reliably	Ensure that the measure m is higher/lower than the value v.	**achieve** miss **with reliability** m [...]
Confidently	Achieve miss and ensure that confidence m is higher/lower than v.	**achieve** miss **with confidence** m [...]
Equidistance	rob performs miss by keeping rob_1 and rob_2 at the same distance.	rob miss **equidistance** rob_1 rob_2
Trail	rob follows object o keeping a distance v.	rob **trail** o **with distance** v

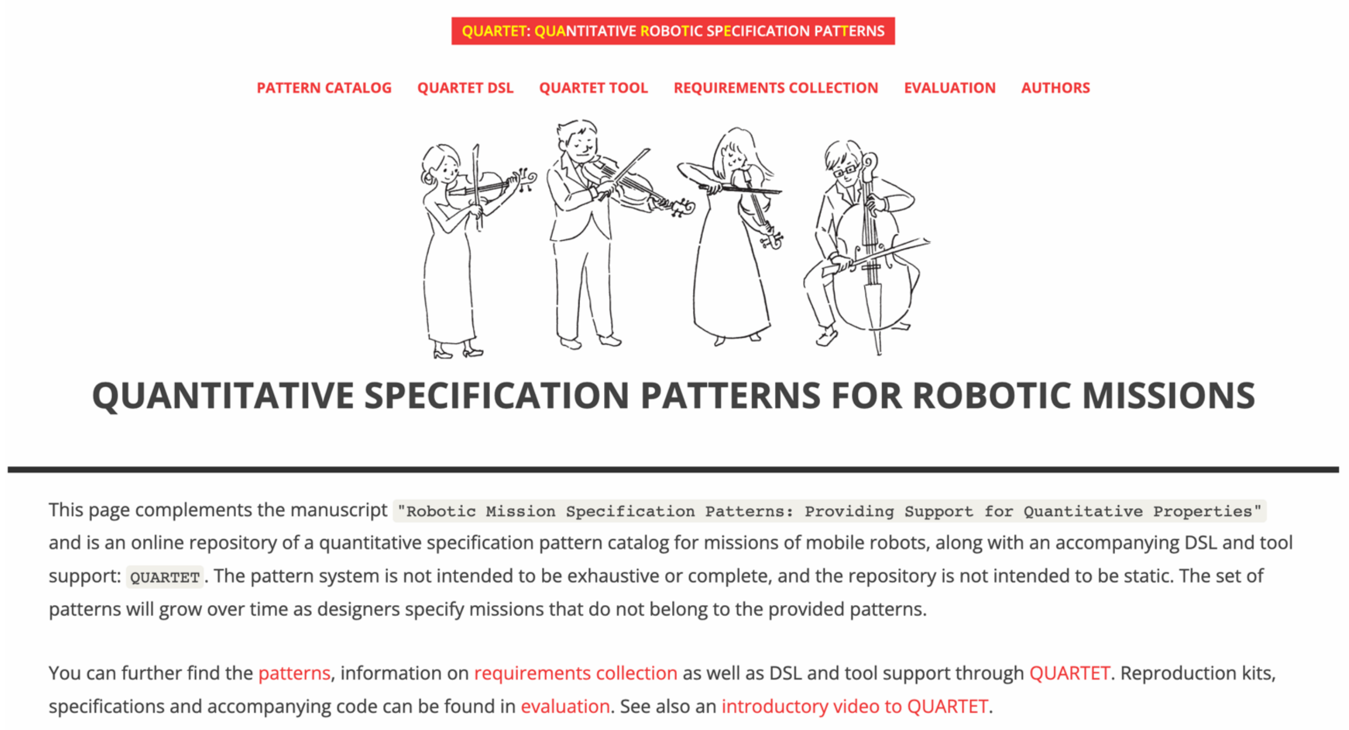
* miss, $miss_1$, $miss_2$ are missions; v, v_1, v_2 are values; rob is a robot, o is an object, m is the name of the quantitative measure.
[...] represents portions of the DSL of Figure 4 omitted for graphical reasons.

Claudio Menghi, Christos Tsigkanos, Mehrnoosh Askarpour , Patrizio Pelliccione, Gricel Vazquez , Radu Calinescu, and Sergio García "Mission Specification Patterns for Mobile Robots: Providing Support for Quantitative Properties," in **IEEE Transactions on Software Engineering (TSE)**, doi: 10.1109/TSE.2022.3230059.

Translation to Probabilistic Reward Computation Tree Logic (PRCTL)

Mission		τ(miss1 **and** miss2)= τ(miss1) $\wedge$ τ(miss2) ; τ(miss1 **or** miss2)= τ(miss1) $\vee$ τ(miss2)
		τ(**not** miss) $= \neg\tau$(miss) ; rob **shall** pat $= \tau$(pat$[r \leftarrow$ rob$]$)
Elementary Patterns	**Prob.**	τ(**maximize** m miss) $= \mathcal{P}_{max=?}(\tau(\text{miss}))$; τ(**minimize** m miss) $= \mathcal{P}_{min=?}(\tau(\text{miss}))$
		τ(m **at most** v miss) $= \mathcal{P}_{\leq v}(\tau(\text{miss}))$; τ(m **less than** v miss) $= \mathcal{P}_{<v}(\tau(\text{miss}))$
		τ(m **at least** v miss) $= \mathcal{P}_{\geq v}(\tau(\text{miss}))$; τ(m **greater than** v miss) $= \mathcal{P}_{>v}(\tau(\text{miss}))$
		τ(m **exactly** v miss) $= \mathcal{P}_{\geq v}(\tau(\text{miss})) \wedge \mathcal{P}_{\leq v}(\tau(\text{miss}))$
		τ(m **within** v_1 **and** v_2 miss) $= \mathcal{P}_{\geq v_1}(\tau(\text{miss})) \wedge \mathcal{P}_{\leq v_2}(\tau(\text{miss}))$
		τ(m **strictly within** v_1 **and** v_2 miss) $= \mathcal{P}_{>v_1}(\tau(\text{miss})) \wedge \mathcal{P}_{<v_2}(\tau(\text{miss}))$
	Rewards	τ(**maximize** m miss) $= \mathcal{E}_{max=?}(\tau(\text{miss}))$; τ(**minimize** m miss) $= \mathcal{E}_{min=?}(\tau(\text{miss}))$
		τ(m **at most** v miss) $= \mathcal{E}_{[0,v]}(\tau(\text{miss}))$; τ(m **less than** v miss) $= \mathcal{E}_{[0,v)}(\tau(\text{miss}))$
		τ(m **at least** v miss) $= \mathcal{E}_{[v,\infty)}(\tau(\text{miss}))$; τ(m **greater than** v miss) $= \mathcal{E}_{(v,\infty)}(\tau(\text{miss}))$
		τ(m **exactly** v miss) $= \mathcal{E}_{\geq v}(\tau(\text{miss})) \wedge \mathcal{E}_{\leq v}(\tau(\text{miss}))$
		τ(m **within** v_1 **and** v_2 miss) $= \mathcal{E}_{[v_1,\infty)}(\tau(\text{miss})) \wedge \mathcal{E}_{[0,v_2]}(\tau(\text{miss}))$
		τ(m **strictly within** v_1 **and** v_2 miss) $= \mathcal{E}_{(v_1,\infty)}(\tau(\text{miss})) \wedge \mathcal{E}_{[0,v_2)}(\tau(\text{miss}))$
Composite Patterns		τ(**conserve** m **while** miss) $= \mathcal{E}_{min=?}(\tau(\text{miss}))$
		τ(**preserve** m **within** $[v_1, v_2]$ **while** miss) $= \mathcal{E}_{[v_1,v_2]}(\tau(\text{miss}))$
		τ(**pause** v miss) $= \mathcal{G}^{[0,v]}\, \tau(\neg\text{miss}) \wedge (\mathcal{F}^{[v+1,v+1]}(\tau(\text{miss})))$
		τ(**timeout** v miss) $= \mathcal{G}^{[v,\infty]}(\neg\tau(\text{miss}))$
		τ(**repeat** miss **every** v) $= \tau(\text{miss}) \wedge \mathcal{G}^{[0,\infty]}(\tau(\text{miss}) \rightarrow (\mathcal{G}^{[1,v-1]}(\neg\tau(\text{miss})) \wedge (\mathcal{F}^{[v,v]}(\tau(\text{miss})))))$
		τ(**end** miss **exactly at** v) $= \mathcal{G}^{[0,v)}(\tau(\text{miss})) \wedge \mathcal{G}^{[v,\infty]}(\neg\tau(\text{miss}))$
		τ(**time of** miss_1 **proportional to** miss_2 **by factor** v)=NA (Not Available in PRCTL)
		τ(**execute** rob **actions** $\text{act}_1, \text{act}_2, \ldots, \text{act}_n$)= $\mathcal{F}\,(\bigwedge_{i=1}^{n} \text{act}_i)$
		τ(r **accrue** m **while** miss)= $\mathcal{E}_{max=?}(\tau(\text{miss}))$
		τ(**achieve** miss **with reliability** m (**greater** | **less**) **than** v)= $\mathcal{E}_{[v,\infty)}(\tau(\text{miss}))/\mathcal{E}_{[0,v)}(\tau(\text{miss}))$
		τ(**achieve** miss **with confidence** m (**greater** | **less**) **than** v)=$\mathcal{L}_{>v}(\tau(\text{miss}))/\mathcal{L}_{<v}(\tau(\text{miss}))$
		τ(rob miss **equidistance** rob_1 rob_2)=NA (Not Available in PRCTL)
		τ(rob **trail** o **with distance** v)=NA (Not Available in PRCTL)

Further info about specification patterns



Claudio Menghi, Christos Tsigkanos, Mehrnoosh Askarpour , Patrizio Pelliccione, Gricel Vazquez , Radu Calinescu, and Sergio García "Mission Specification Patterns for Mobile Robots: Providing Support for Quantitative Properties," in **IEEE Transactions on Software Engineering (TSE)**, doi: 10.1109/TSE.2022.3230059.

https://roboticpatterns.com/quantitative

https://github.com/Gricel-lee/Quartet-MRS-DSL

Mission specification patterns for robots simplify and makes the specification accessible.

What’s missing?

G S S I

Mission specification patterns for robots simplify and makes the specification accessible.

What’s missing?

- Single robots
- Focus on movements
- How to deal with variability of the real world?

GS
SI

Two steps

What kind of missions are specified in practice?

- Identification of missions already specified in practice (i.e. papers, documents of robotic companies)
- Definition of a catalogue of mission specification patterns
- Tool support for assisting users in the specification of missions via the use and instantiation of patterns

How to use these patterns to specify complex missions?

- Definition of operators to combine the mission specification patterns
- Definition of a Domain Specific Language (DSL) with graphical and textual syntax
- Definition of a tool support for the DSL

Two steps

patterns

What kind of missions are specified in practice?

- Identification of missions already specified in practice (i.e. papers, documents of robotic companies)
- Definition of a catalogue of mission specification patterns
- Tool support for assisting users in the specification of missions via the use and instantiation of patterns

How to use these patterns to specify complex missions?

- Definition of operators to combine the mission specification patterns
- Definition of a Domain Specific Language (DSL) with graphical and textual syntax
- Definition of a tool support for the DSL

Two steps

What kind of missions are specified in practice?

- Identification of missions already specified in practice (i.e. papers, documents of robotic companies)
- Definition of a catalogue of mission specification patterns
- Tool support for assisting users in the specification of missions via the use and instantiation of patterns

How to use these patterns to specify complex missions?

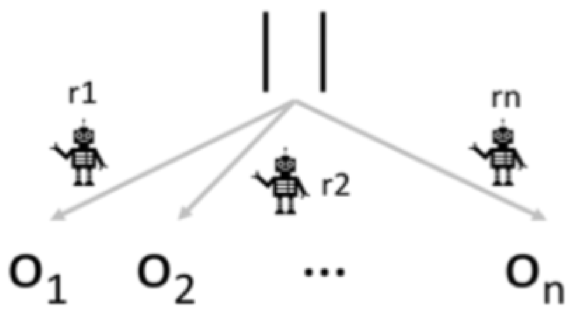
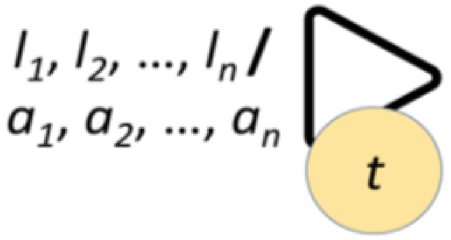
- Definition of operators to combine the mission specification patterns
- Definition of a Domain Specific Language (DSL) with graphical and textual syntax
- Definition of a tool support for the DSL

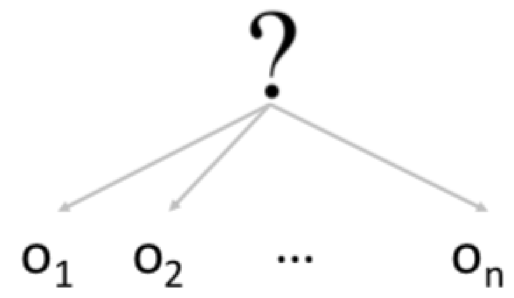
Domain Specific Language to specify missions

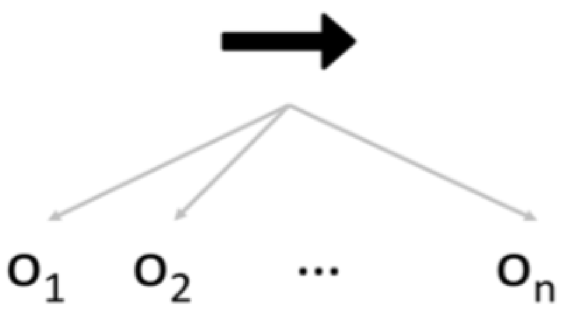
- PROMISE (simPle RObot MIssion SpEcification)
  - Patterns are basic building blocks
  - Operators enable the composition of patterns towards the specification of complex missions for multi-robots

Sergio Garcia, Patrizio Pelliccione, Claudio Menghi, Thorsten Berger, Tomas Bures, "Higher-Level Mission Specification for Multiple Robots," in *12th ACM SIGPLAN International Conference on Software Language Engineering (SLE)*, 2019.

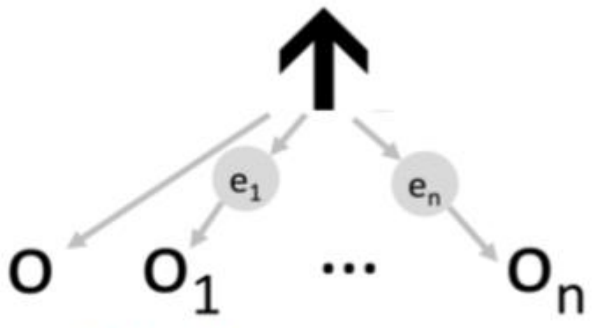
Operators of the DSL

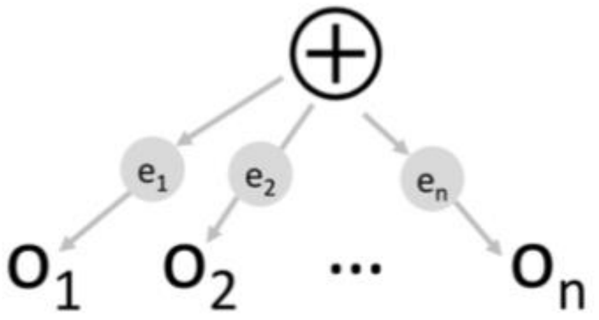
Name	Description	Semantics	Syntax	Intermediate language
Parallel $\|\|(r_1,\ldots r_n, o_1,\ldots,o_n)$	Always the root of the mission. The operators $o_1, o_2, \cdots, o_n$ are executed in parallel, each by a different robot—i.e., assigns one branch to each robot. Returns success when all operators return success, failure otherwise.	$\{res_1, res_2, \cdots, res_n\}=\{o_1, o_2, \cdots, o_n\}$ **if** $(res_1 == \top \wedge \cdots \wedge res_n == \top)$ **then** **return** $\top$ **else return** $\bot$	 parallel$\{r1(o_1), \ldots, rn(o_n)\}$	r1[**o1**] r2[**o2**] ... rn[**on**]
Delegate $\triangleright(\mathcal{E}, t)$	Delegates execution of a task t to a specific robot (specified by the Parallel operator). Tasks are specified using patterns for robotic missions that take as input parameters as locations (indicated as $l_1, l_2, \ldots, l_n$) and actions (indicated as $a_1, a_2, \ldots, a_n$).	**execute**$(\mathcal{E}, t)$	 delegate(t locations $l_1, \ldots, l_n$) delegate(t actions $a_1, \ldots, a_n$)	LTL formula of the pattern specified by the task t.

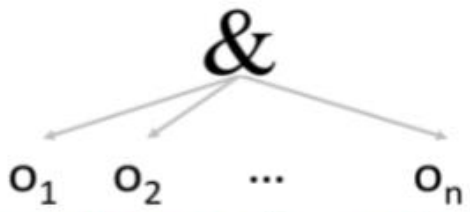
Fallback $?(\{o_1, o_2, \cdots, o_n\})$	Executes the first operator; if it is executed successfully, ends with success. If the execution of the first operator fails, tries to execute the second operator. This procedure is repeated for all the other operators. Returns failure if all operators fail.	**if** $(\{o_1,o_2,\cdots,o_n\} \neq \emptyset)$ **then**
		$res = o_1$;
		if$(res == \bot)$ **then**
		$?(\{o_2, \cdots, o_n\})$
		else return $\top$
		else return $\bot$
			 fallback $(o_1, o_2, \ldots, o_n)$	parent[**fb**]
				fb_1[**o1**]
				fb_2[**o2**]
				...
				fb_n[**on**]

Sequence $\rightarrow(\{o_1, o_2, \cdots, o_n\})$	Executes all the operators from the first to the last. If an operator returns success executes the subsequent operator. If an operator returns a failure returns failure. Returns success if and only if all the operators return success.	**if**$(\{o_1, o_2, \cdots, o_n\} \neq \emptyset)$ **then**
		$res = o_1$;
		if$(res == \top)$ **then**
		$\rightarrow(\{o_2, \cdots, o_n\})$
		else return $\bot$
		else return $\bot$
			 sequence $(o_1, o_2, \ldots, o_n)$	[**o1**,**o2**,...,**on**]

Operators of the DSL

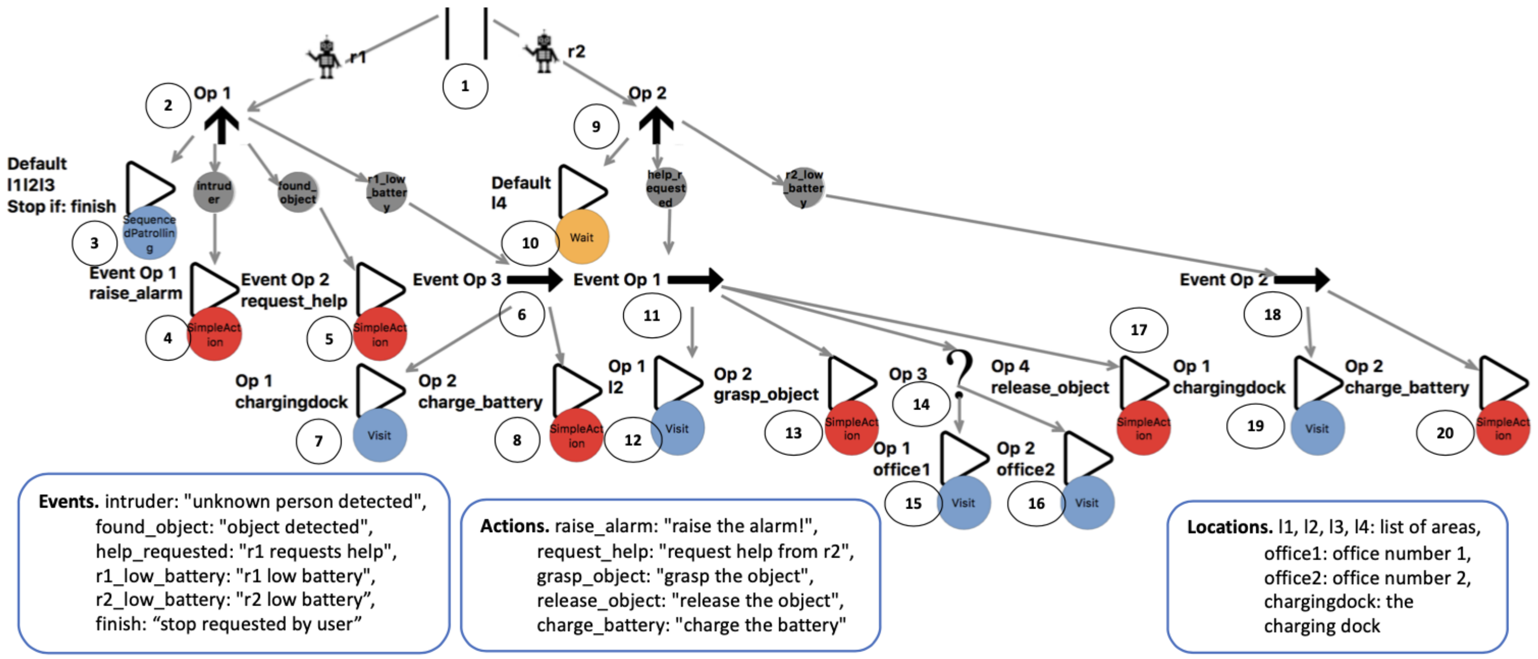
EventHandler
$\Uparrow(e_1, \ldots, e_n, o, o_1, \ldots, o_n)$
	Executes a by default operator o. Once an event e_i occurs, executes operator o_i in response. Once the execution of o_i is finished, resumes the operator o. Returns success if the operator o succeeds and all the events that occurred during the execution of o are correctly handled.	$res = \bot$;
		while($res \neq \top$)
		  $res = o$;
		  if($res == \top$) **then**
		    return $\top$
		  if($e_i == \top$), **then** $i = 1, \ldots, n$
		    $resint = o_i$;
		    if($resint == \bot$), **then**
		      return $\bot$
		    $res =$ **resume**(o);
		return res
			
			eventHandler(
			default(o)
			except e_1 (o_1)
			except e_2 (o_2)...
			except e_n (o_n))
				parent[**eh**]
				eh_default[**o**]
				eh_e1[**o1**]
				eh_e2[**o2**]
				...
				eh_en[**on**]

Condition
$\oplus(\{e_1, \cdots, e_n, o_1, \cdots, o_n\})$
	Evaluates the conditions from the first to the last. If the evaluation of one or more conditions is true, executes the corresponding operators. Returns $\bot$ if an operation is not successful, i.e., either it fails or an event occurs. Returns $\top$ when all the executed operations return $\top$.	**if**($e_1 == \top$) *then*
		  $res = o_1$
		  if($res == \bot$) **then**
		    return $\bot$
		...
		if($e_n == \top$) *then*
		  $res = o_n$
		  if($res == \bot$) **then**
		    return $\bot$
		return $\top$
			
			condition(
			if e_1 then (o_1)
			if e_2 then (o_2)...
			if e_n then (o_n))
				parent[**cond**]
				cond_e1[**o1**]
				cond_e2[**o2**]
				...
				cond_en[**on**]

TaskComb. $\&(\{o_1, o_2\})$	Allows the composition of a *core movement* task with one or more *avoidance* tasks and with one or more *trigger* tasks. The composition is performed by means of the *and* logical operator.	$res = o_1 \&\& o_2 \&\& \ldots o_n$ **if**($res == \top$) **then** **return** $\top$ **else return** $\bot$	 combination(o_1 and o_2 and ... o_n)	[**o1** && **o2** && ... **on**]

```
Mission:
 'mission' '{'
 ('conditions' '{' ('events' events+=Event ( "," events+=Event)*)?
 ('actions' actions+=Action ( "," actions+=Action)*)? '}')?
 'robots' robots+=Robot ( "," robots+=Robot)*
 ('locations' locations+=Location ( "," locations+=Location)*)?
 'operators' '{' operator+=Operator ( "," operator+=Operator)* '}'
 '}';
Operator:
 FallBackOp | SequenceOp | ParallelOp | EventHandlerOp |
 ConditionOp | DelegateOp | TaskCombinationOp;
Tasks:
 //List of tasks from the provided catalog
Robot:
 name=EString;
Location:
 name=EString;
Event:
 name=ID ':' description=EString;
Action:
 name=ID ':' description=EString;
FallBackOp:
 'fallback' '(' inputOperators+=Operator
 ("," inputOperators+=Operator)* ')';
SequenceOp:
 'sequence' '(' inputOperators+=Operator
 ("," inputOperators+=Operator)*')';
ParallelOp:
 'parallel'
 '{'(inputRobots+=[Robot|EString] '(' inputOperators+=Operator')'
 ("," inputRobots+=[Robot|EString] '('inputOperators+=Operator')')
      *)?'}';
EventHandlerOp:
 'eventHandler' '('
 'default' '(' inputOperators+=Operator ')'
 ('except' inputEvents+=EventAssignment)+')';
ConditionOp:
 'condition' '('
 ('if' inputEvents+=EventAssignment )+')';
TaskCombinationOp:
 'combination' '(' inputOperators+=Operator
 (('&' | 'AND' | 'and') inputOperators+=Operator)+ ')';
DelegateOp:
 'delegate' '(' task=Tasks
 ('locations' inputLocations+=[Location|EString]
 ("," inputLocations+=[Location|EString])*  )?
 ('actions' inputAction+=[Action|EString]
 ( "," inputAction+=[Action|EString])*)?
 ('stoppingEvents' stoppingEvent+=[Event|EString]
 ( "," stoppingEvent+=[Event|EString])*)? ')';
EventAssignment:
 inputEvent=[Event|EString] '(' inputOperators=Operator ')';
```

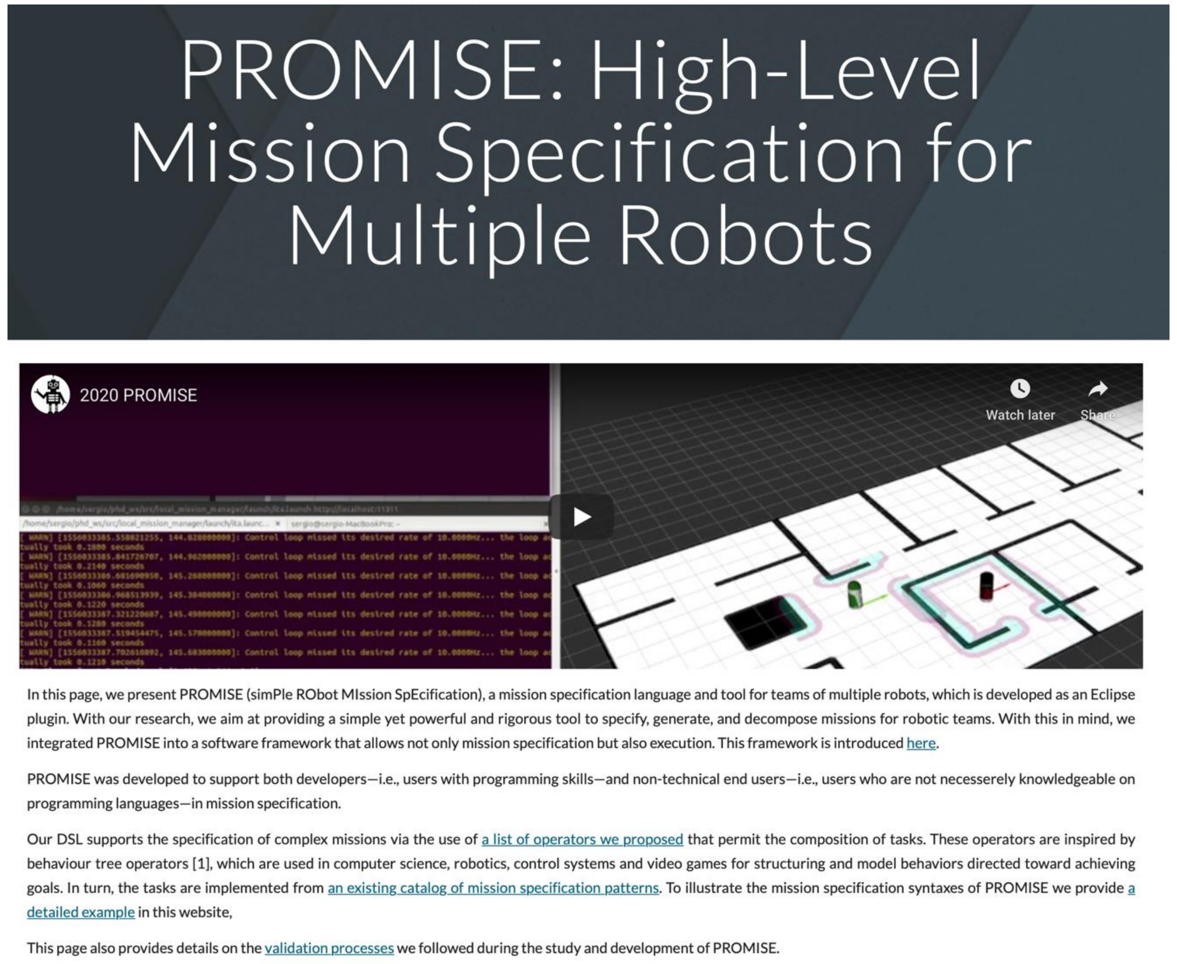
Grammar
(Abstract syntax)




```
operators{ parallel{ (1)
  r1(eventHandler( (2)
    default(delegate ( SequencedPatrolling locations l1, l2, l3 (3)
      stoppingEvents finish))
    except intruder (delegate (SimpleAction actions raise_alarm)) (4)
    except found_object (delegate (SimpleAction actions request_help)) (5)
    except r1_low_battery (sequence( (6)
      delegate(Visit locations chargingdock), (7)
      delegate(SimpleAction actions charge_battery))))), (8)
  r2(eventHandler( (9)
    default(delegate(Wait locations l4)) (10)
    except help_requested( sequence( (11)
      delegate(Visit locations l2), (12)
      delegate(SimpleAction actions grasp_object), (13)
      fallback ( (14)
        delegate(Visit locations office1), (15)
        delegate(Visit locations office2)), (16)
        delegate ( SimpleAction actions release_object ) ) ) (17)
    except r2_low_battery(sequence( (18)
      delegate(Visit locations chargingdock), (19)
      delegate (SimpleAction actions charge_battery))))) (20)
    }
  }
```

Two concrete syntaxes
(Graphical – behaviour tree style - and Textual)

Further info about Promise



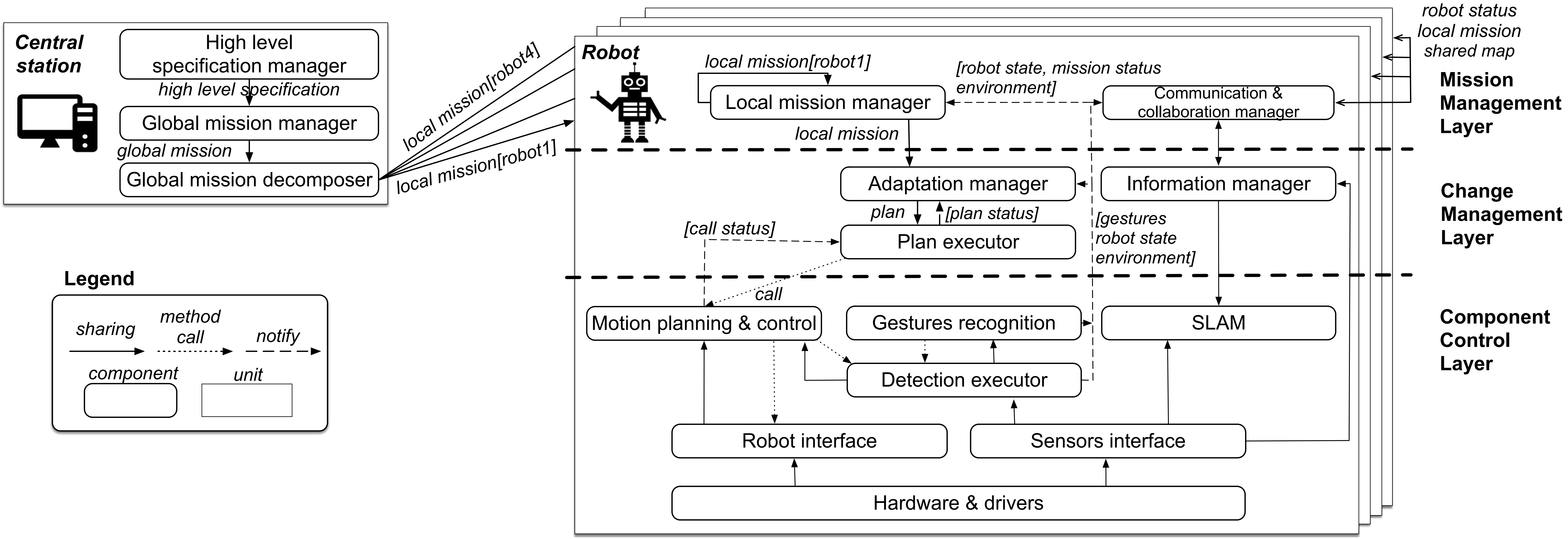
https://sites.google.com/view/promise-dsl/home

García, S., Pelliccione, P., Menghi, C., Berger, T., & Bures, T. (2019, October). High-level mission specification for multiple robots. In Proceedings of the 12th ACM SIGPLAN International Conference on **Software Language Engineering (SLE)** (pp. 127-140). ACM.

García, S., Pelliccione, P., Menghi, C., Berger, T., & Bures, T. (2020,). PROMISE: High-Level Mission Specification for Multiple Robots. In 2nd International Conference on Software Engineering Companion (**ICSE ’20 Demo**).

https://github.com/SergioGarG/PROMISE_implementation

SERA (Self- adaptive dEcentralized Robotic Architecture)



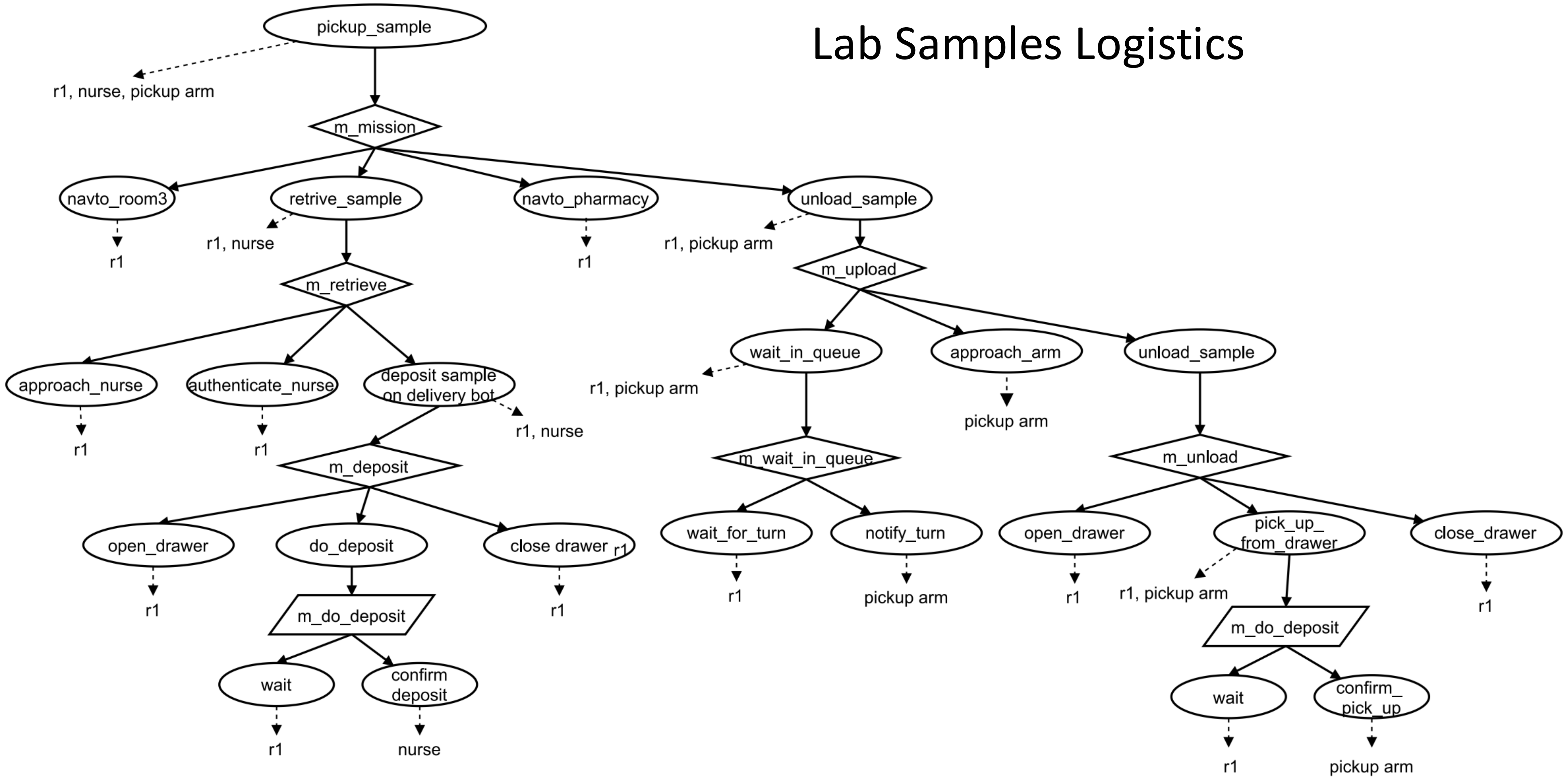
https://co4robots.eu/

S. García, C. Menghi, P. Pelliccione, T. Berger and R. Wohlrab, "An Architecture for Decentralized, Collaborative, and Autonomous Robots," *2018 IEEE International Conference on Software Architecture (ICSA)*, Seattle, WA, 2018, pp. 75-7509.

Instantiated Hierarchical Task Networks (iHTN)

- Hierarchical Task Networks is a formalism for task planning.
- The Instantiated HTN (iHTN) formalism formalizes a multi-robot collaborative mission.

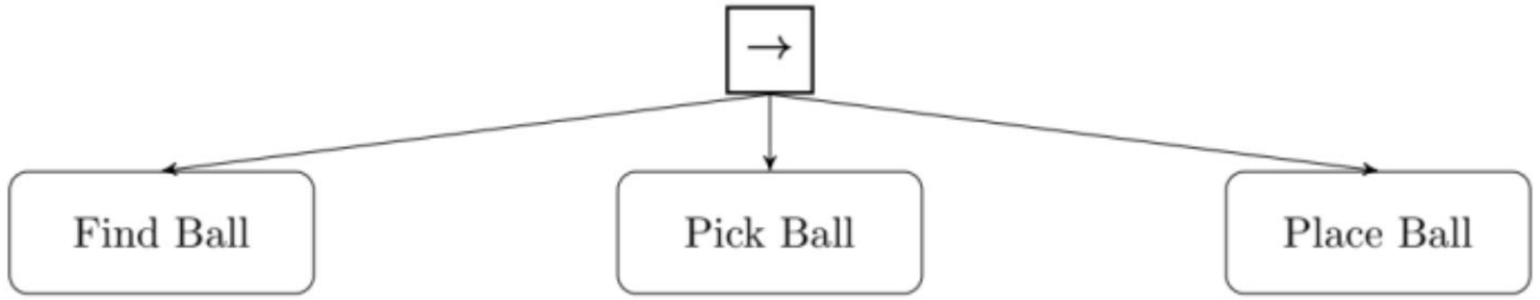
- Tasks (ellipses) are efforts that a set of agents (robots or humans) must undertake.
- A task can be abstract or concrete.
- Abstract tasks are refined by methods.
- Methods are linked to tasks of a lower level and a type of ordering.
- The ordering can be sequential (diamond) or unordered (parallelogram).
- …



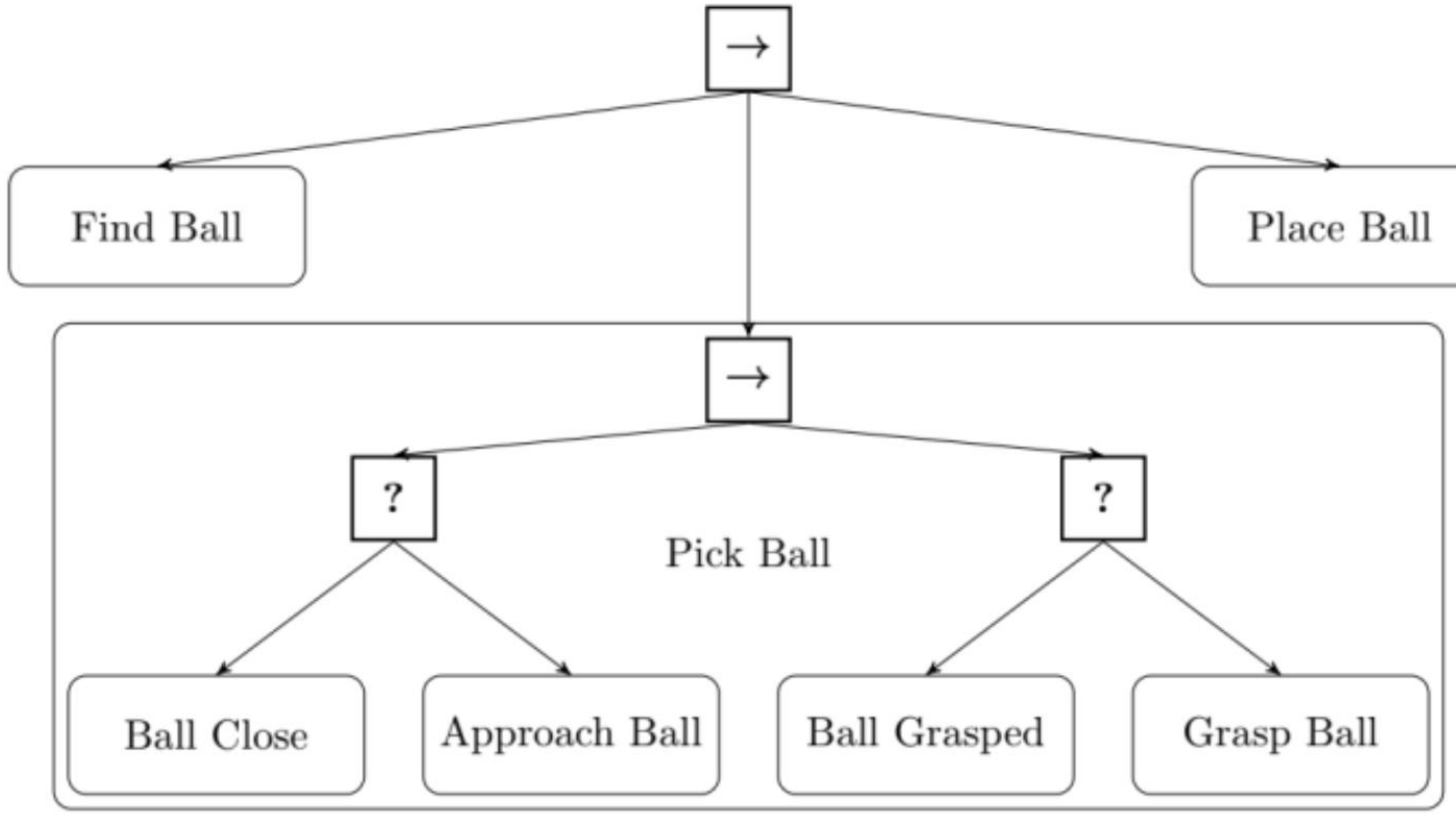
Lab Samples Logistics

Gabriel Rodrigues, Ricardo Caldas, Gabriel Araujo, Vicente de Moraes, Genaína Rodrigues, Patrizio Pelliccione, An architecture for mission coordination of heterogeneous robots, Journal of Systems and Software, 191, 2022, https://doi.org/10.1016/j.jss.2022.111363

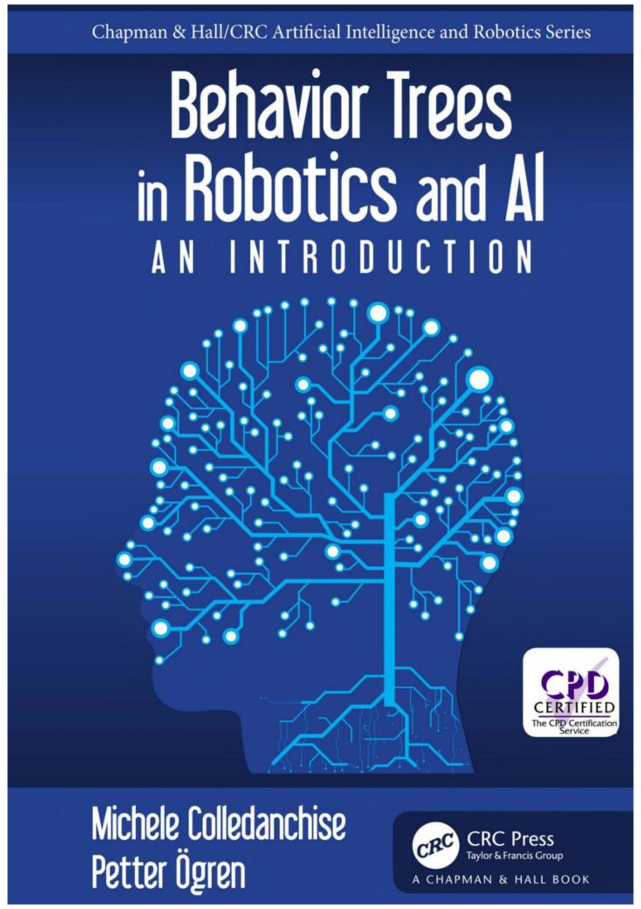
Behavior trees



(a) A high level BT carrying out a task consisting of first finding, then picking and finally placing a ball.



(b) The Action Pick Ball from the BT in Figure 1.1(a) is expanded into a sub-BT. The Ball is approached until it is considered close, and then the Action Grasp is executed until the Ball is securely grasped.

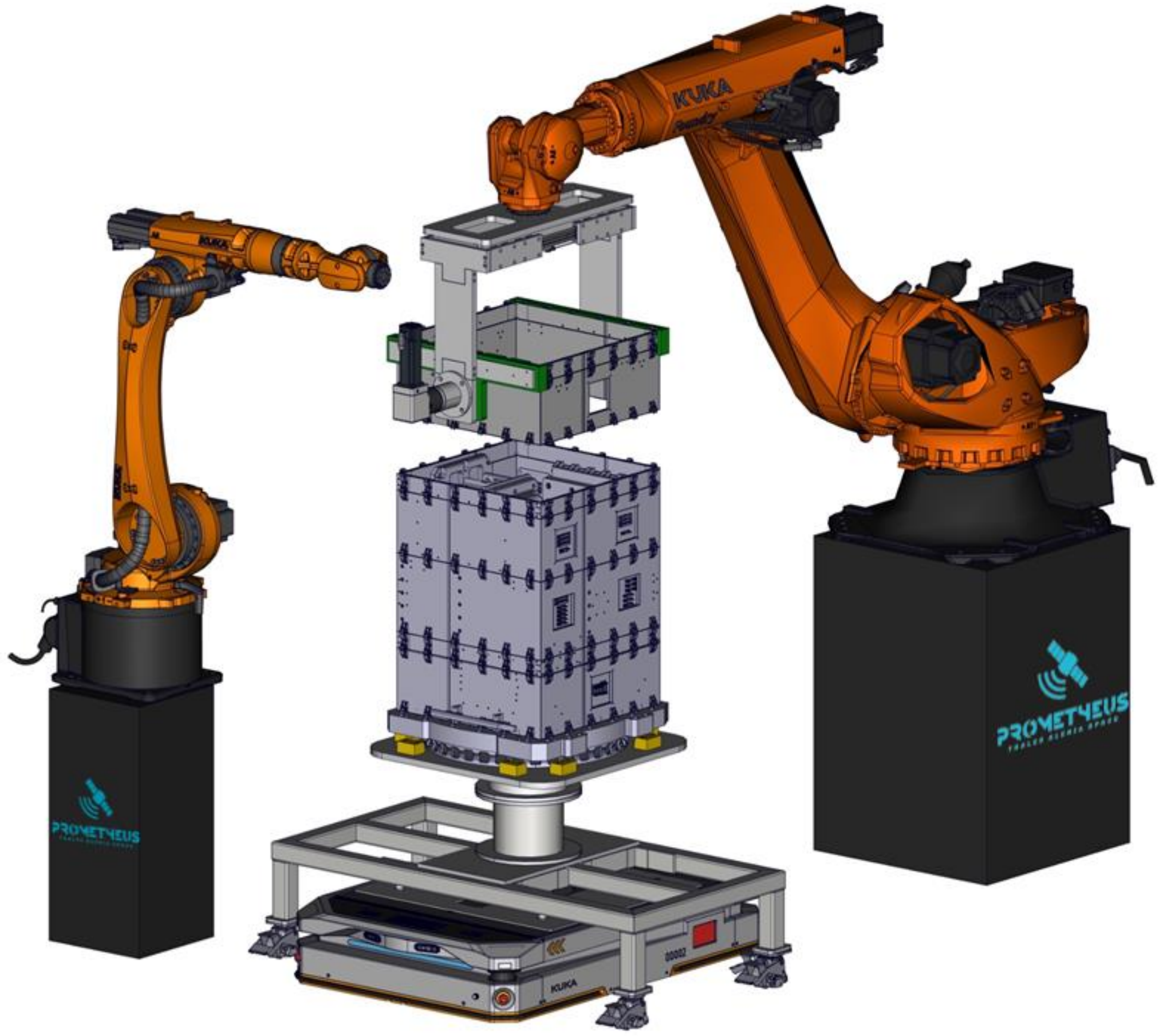


Node type	Symbol	Succeeds	Fails	Running
Fallback	?	If one child succeeds	If all children fail	If one child returns Running
Sequence	→	If all children succeed	If one child fails	If one child returns Running
Parallel	⇉	If $\geq M$ children succeed	If $> N - M$ children fail	else
Action	text	Upon completion	If impossible to complete	During completion
Condition	text	If true	If false	Never
Decorator	◊	Custom	Custom	Custom

Democratizing the programming and use of Industrial Robots

Democratization of Robot Engineering for Advanced Manufacturing (manufacturing satellites)

- Accessible by users without expertise in ICT or robotic
- Coordination of multi and heterogeneous robots and human operators
- It forces modularity and programming with reuse (parametric APIs)



Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ROME TECHNOPOLE
Fondazione Rome Technopole

Domain Specific Language components

Agents (Robots & Operators):
- Robot1, Robot2, HumanOP1, AMR1

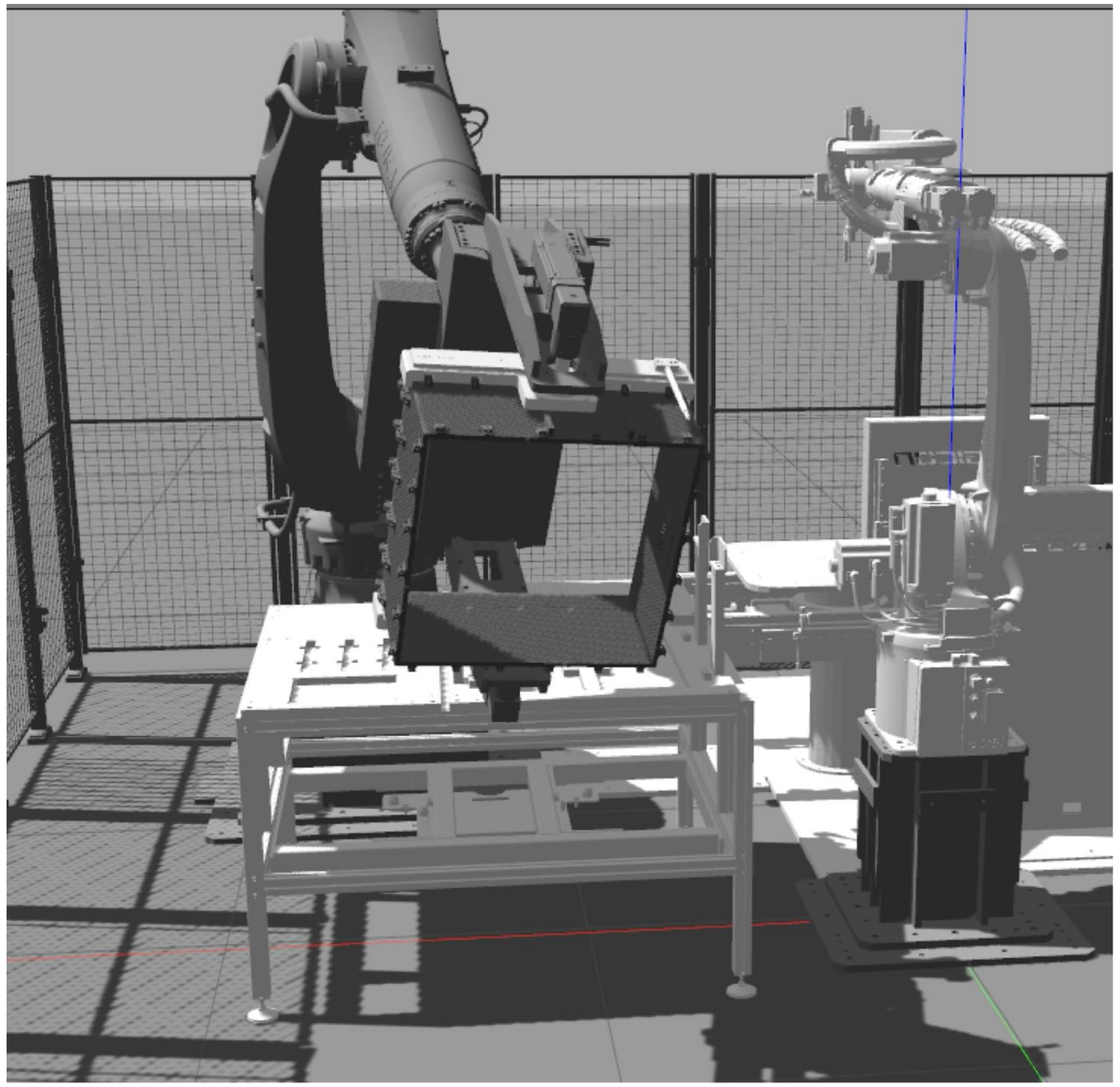
Locations:
- Quality Control, Assembly Location, Warehouse, Stacking Platform, Buffer Area

Trays & Components:
- AOCS Tray, DHC Tray, Components (CMG, MTQ, MAG, screws)

Mission Tasks:
- Moving, Picking, Placing, Screwing, Assembling

Workflow

- Domain Specific Language
- Interpreter (RobotManager)
- Function Calls (Custom interface)
- ROS
- Simulation/Robots



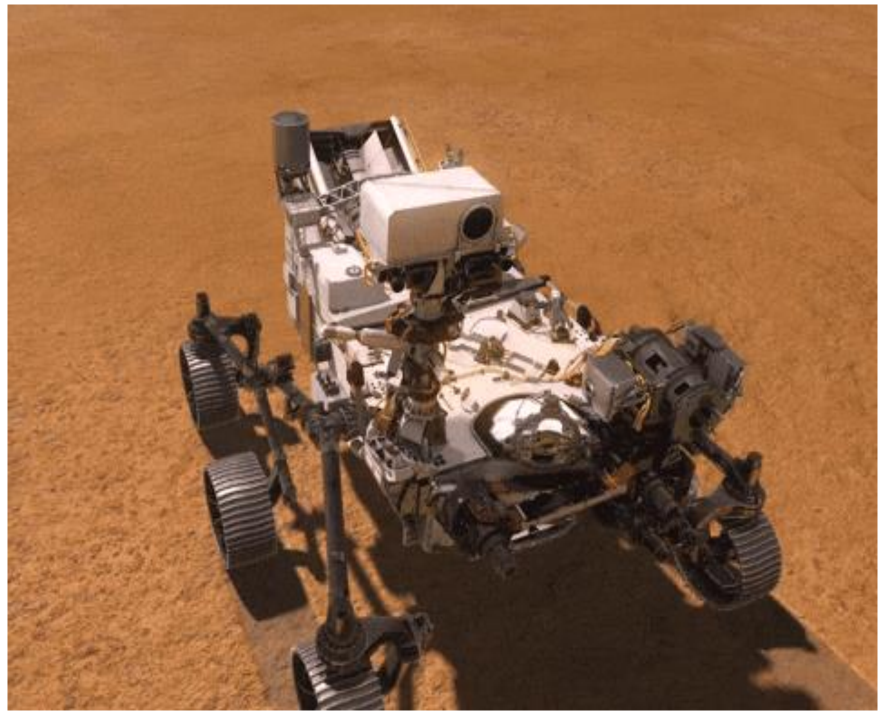
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
Fondazione Rome Technopole

Video

What’s next?

Anomalous and premature wheel wear

- Caused by sharp rocks in Mars terrain
- Wheel design was made according to the current knowledge



Engineers adapted navigation to solve the issue

- Different navigation for different terrains
- Required a software patch

NASA's MSL "Curiosity" rover issues

Lack of precise and complete knowledge at design time: (a face of) uncertainty

GSSI

Dealing with uncertainty in robotic missions

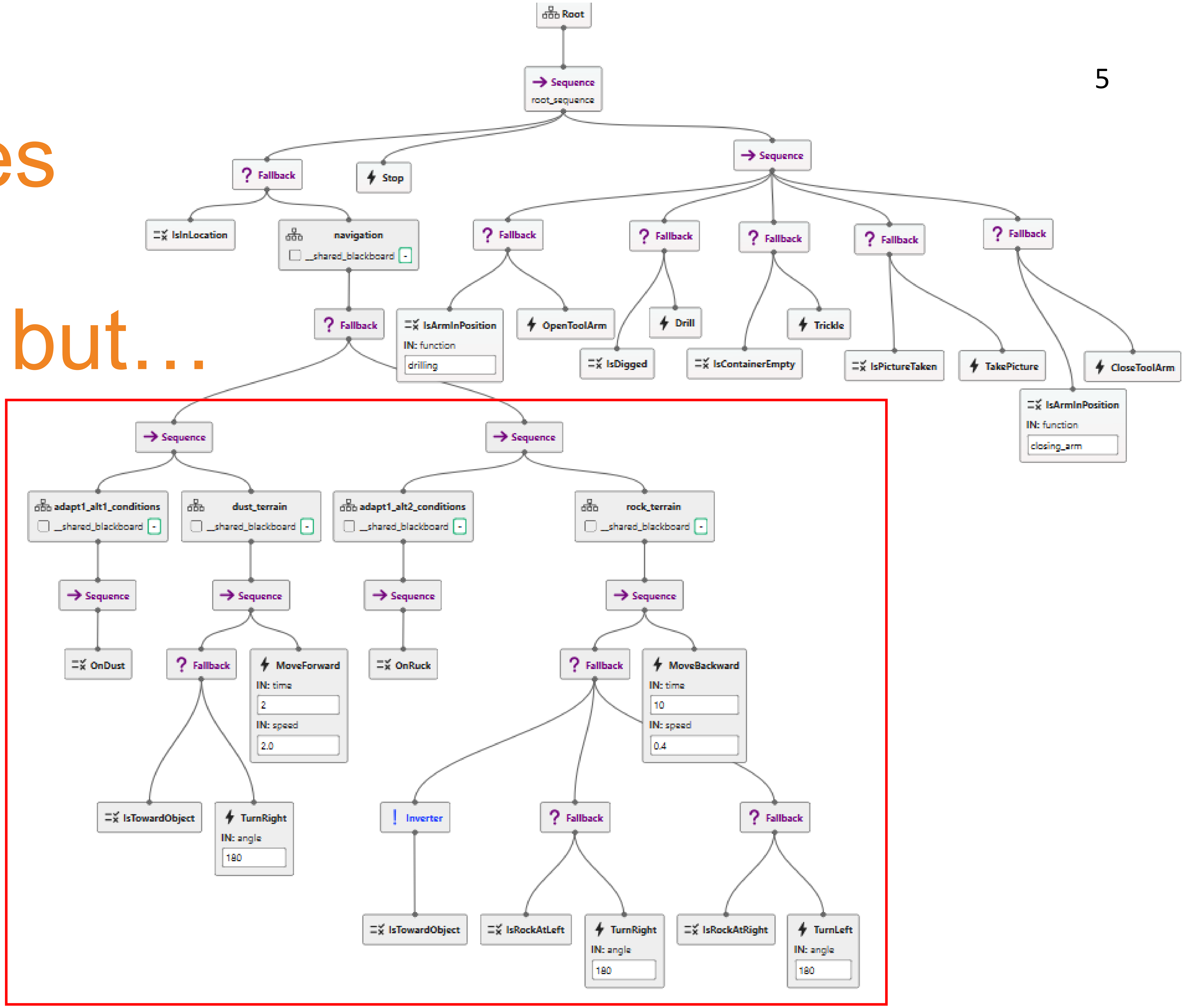
Effects of events/conditions may be unknown at runtime

Self-adaptation is needed to handle uncertainties at runtime

Impractical to specify all the alternative behaviors in a unique model

- Sometimes it impossible (they are not known!)

Behavior Trees enable reactiveness, but…



Adaptable & Uncertainty-aware BTs

Introducing *adaptable nodes*

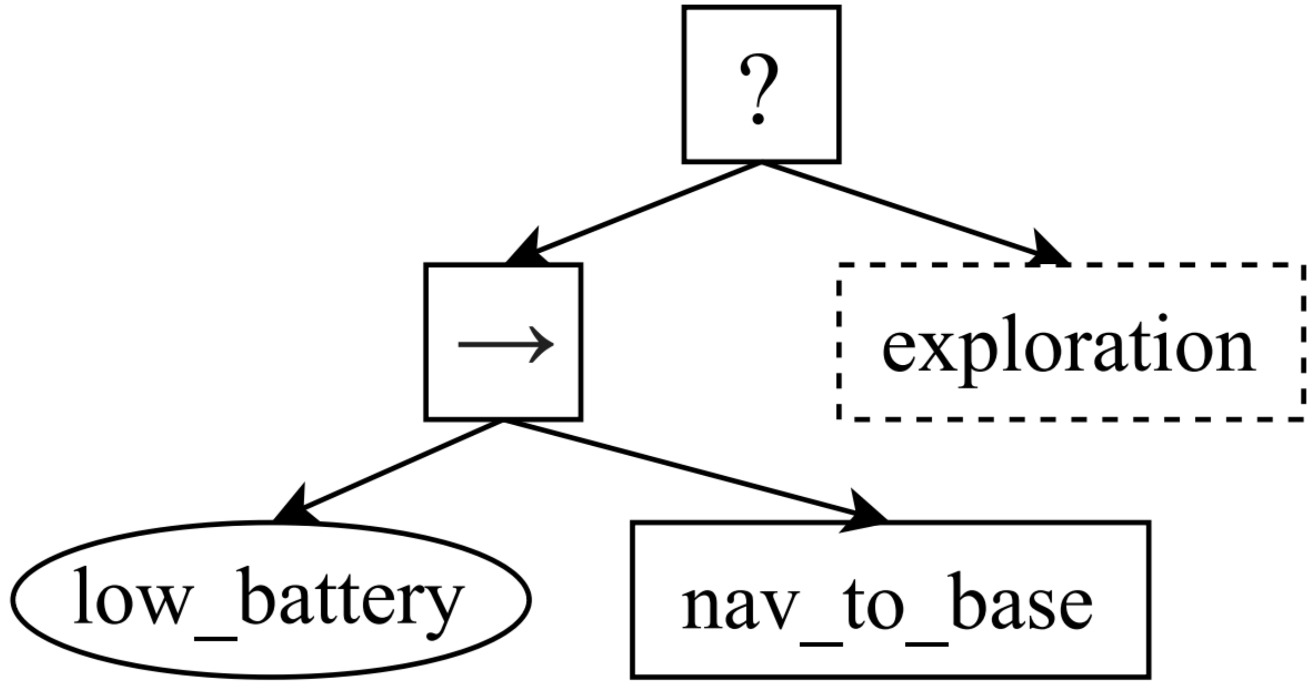
- Abstract nodes that model points of uncertainty
- Manage *known-unknowns*
- Placeholder for alternatives

Goals

- Avoid hard-coding of the alternatives
- Increase modularity and flexibility
- Allow behaviors addition/update

What's missing?

- Specification of the conditions for alternatives
- Runtime support



Adaptable BT

What about the unknown unknown?

Can LLMs help on that?

Domain Specific Languages

Are the patterns domain specific?

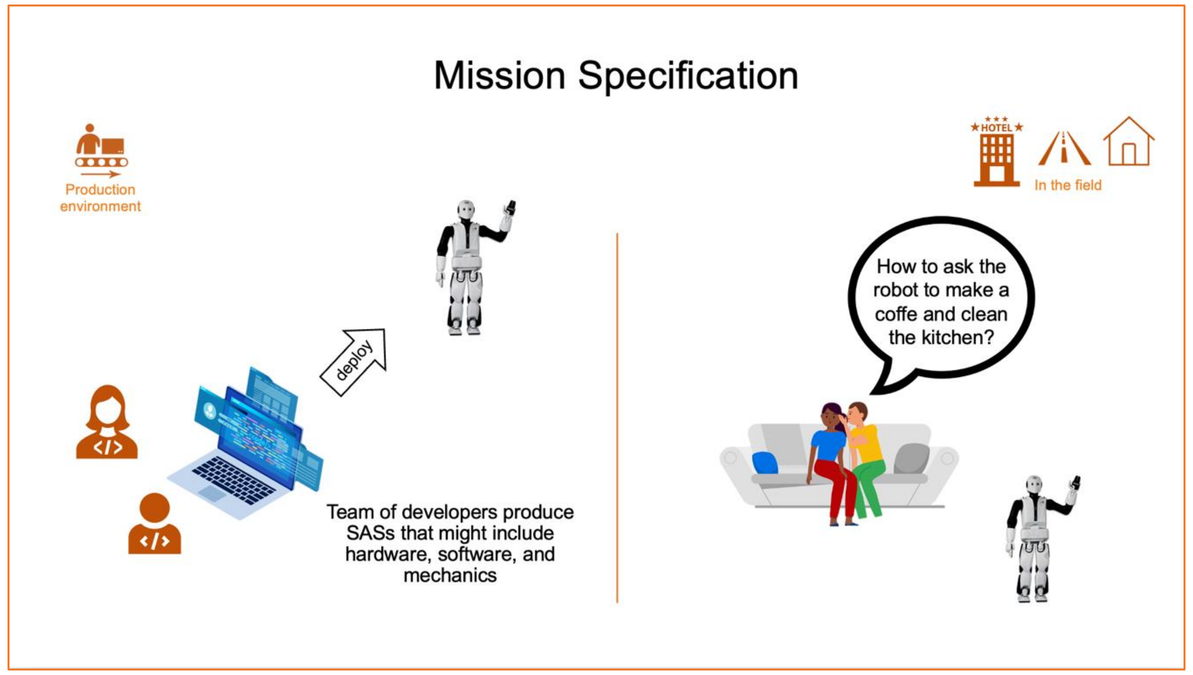
We are not reusing them for the Smart factory

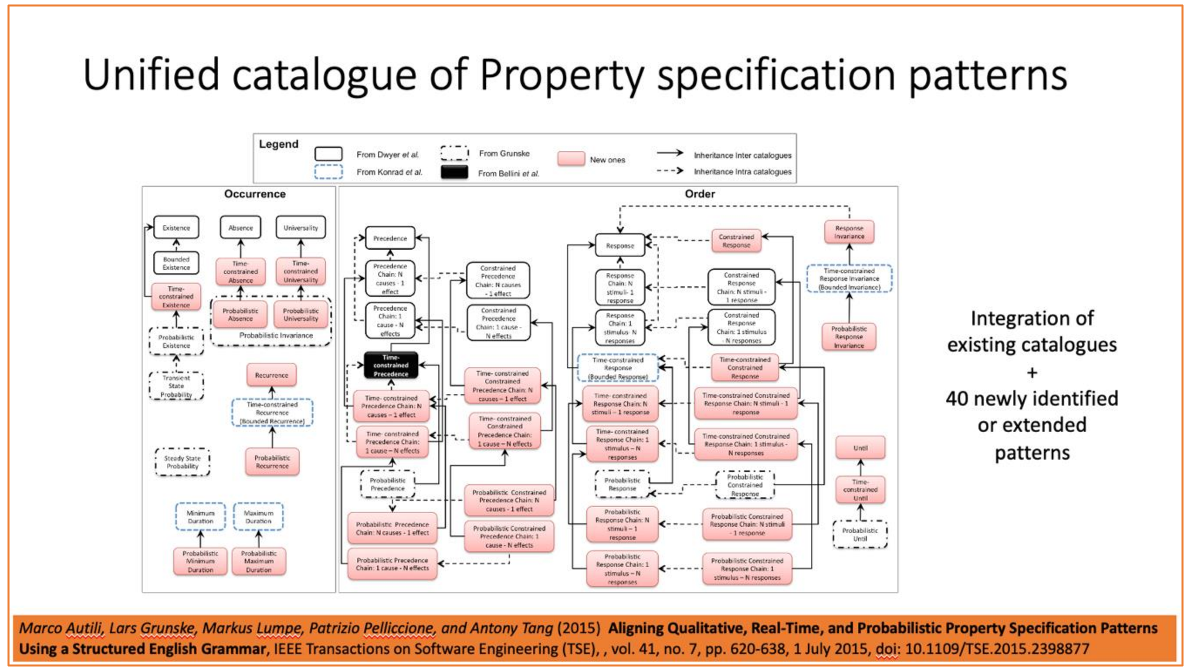
We probably need another step of abstraction in specific domains, like agriculture, space exploration, manufacturing

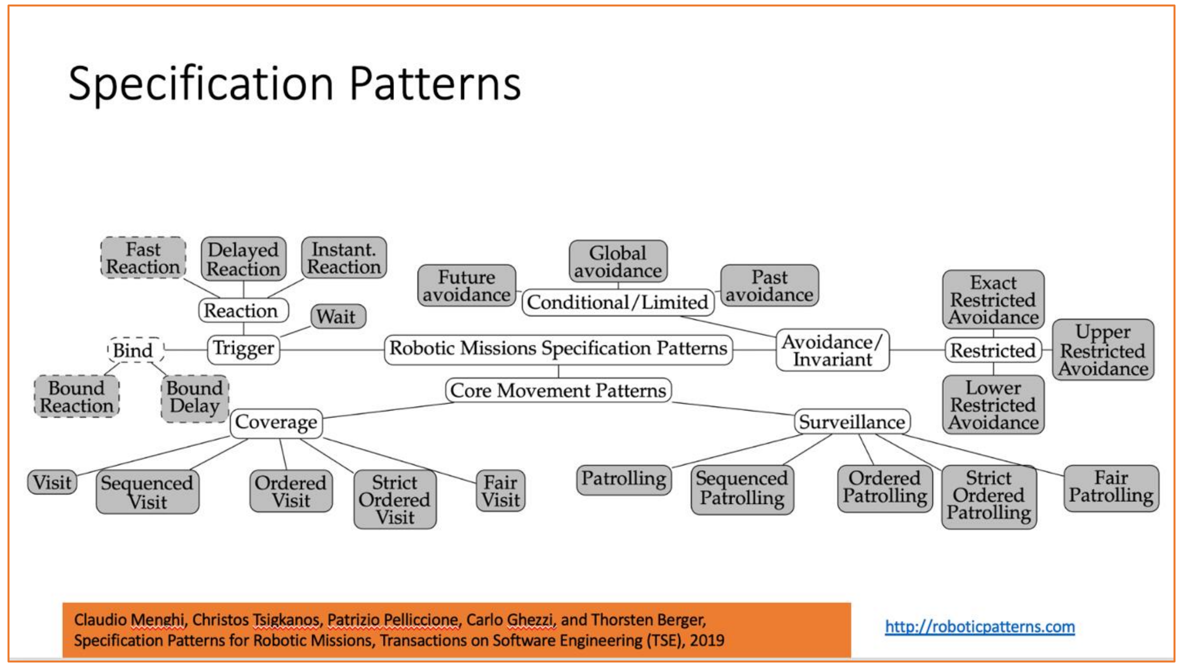
…plus patterns not focusing only on movements (in a map)

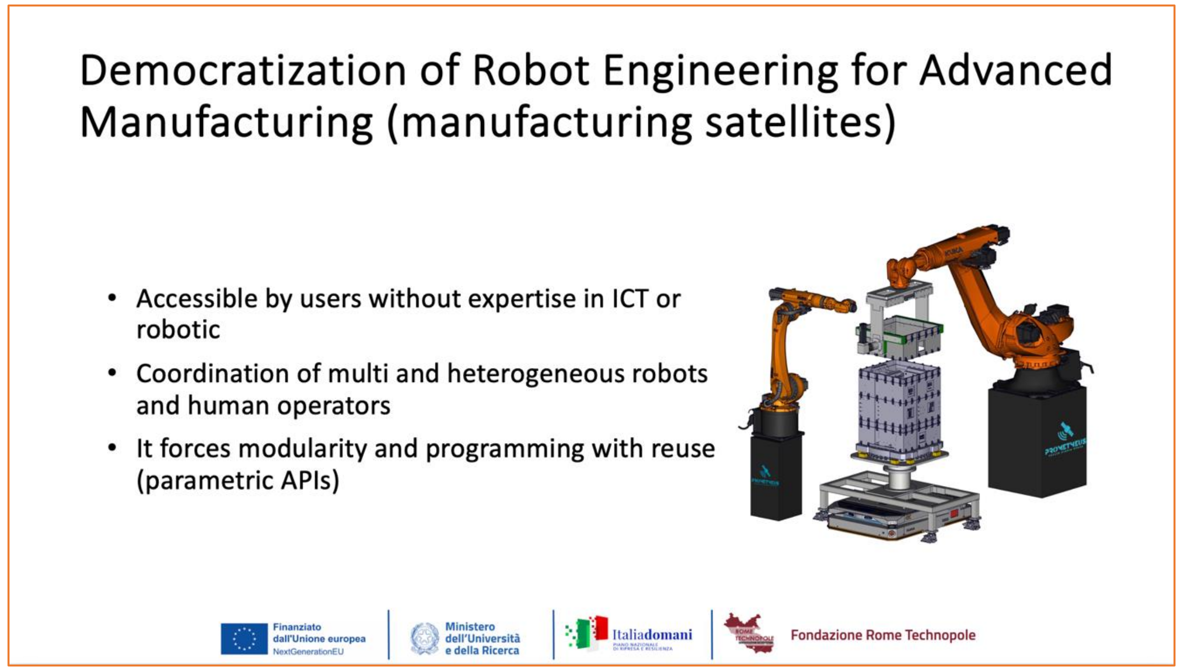
G S
S I

Summary


Mission Specification
Production environment
deploy
Team of developers produce SASs that might include hardware, software, and mechanics
In the field
How to ask the robot to make a coffe and clean the kitchen?


Unified catalogue of Property specification patterns
Legend
Occurrence
Order
Integration of existing catalogues
+
40 newly identified or extended patterns
Marco Autili, Lars Grunske, Markus Lumpe, Patrizio Pelliccione, and Antony Tang (2015) Aligning Qualitative, Real-Time, and Probabilistic Property Specification Patterns Using a Structured English Grammar, IEEE Transactions on Software Engineering (TSE), , vol. 41, no. 7, pp. 620-638, 1 July 2015, doi: 10.1109/TSE.2015.2398877


Specification Patterns
Fast Reaction
Delayed Reaction
Instant. Reaction
Reaction
Wait
Bind
Trigger
Bound Reaction
Bound Delay
Future avoidance
Global avoidance
Past avoidance
Conditional/Limited
Robotic Missions Specification Patterns
Avoidance/ Invariant
Exact Restricted Avoidance
Restricted
Upper Restricted Avoidance
Lower Restricted Avoidance
Core Movement Patterns
Coverage
Surveillance
Visit
Sequenced Visit
Ordered Visit
Strict Ordered Visit
Fair Visit
Patrolling
Sequenced Patrolling
Ordered Patrolling
Strict Ordered Patrolling
Fair Patrolling
Claudio Menghi, Christos Tsigkanos, Patrizio Pelliccione, Carlo Ghezzi, and Thorsten Berger, Specification Patterns for Robotic Missions, Transactions on Software Engineering (TSE), 2019
http://roboticpatterns.com


Democratization of Robot Engineering for Advanced Manufacturing (manufacturing satellites)
Accessible by users without expertise in ICT or robotic
Coordination of multi and heterogeneous robots and human operators
It forces modularity and programming with reuse (parametric APIs)
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
Fondazione Rome Technopole

GRAN SASSO SCIENCE INSTITUTE

Patrizio Pelliccione

patrizio.pelliccione@gssi.it
http://www.patriziopelliccione.com/

www.gssi.it