CONVINCE

Live Autonomous System Developer Survey

60
Total Responses

60 Completed Responses

0 Partial Responses

466
Survey Visits

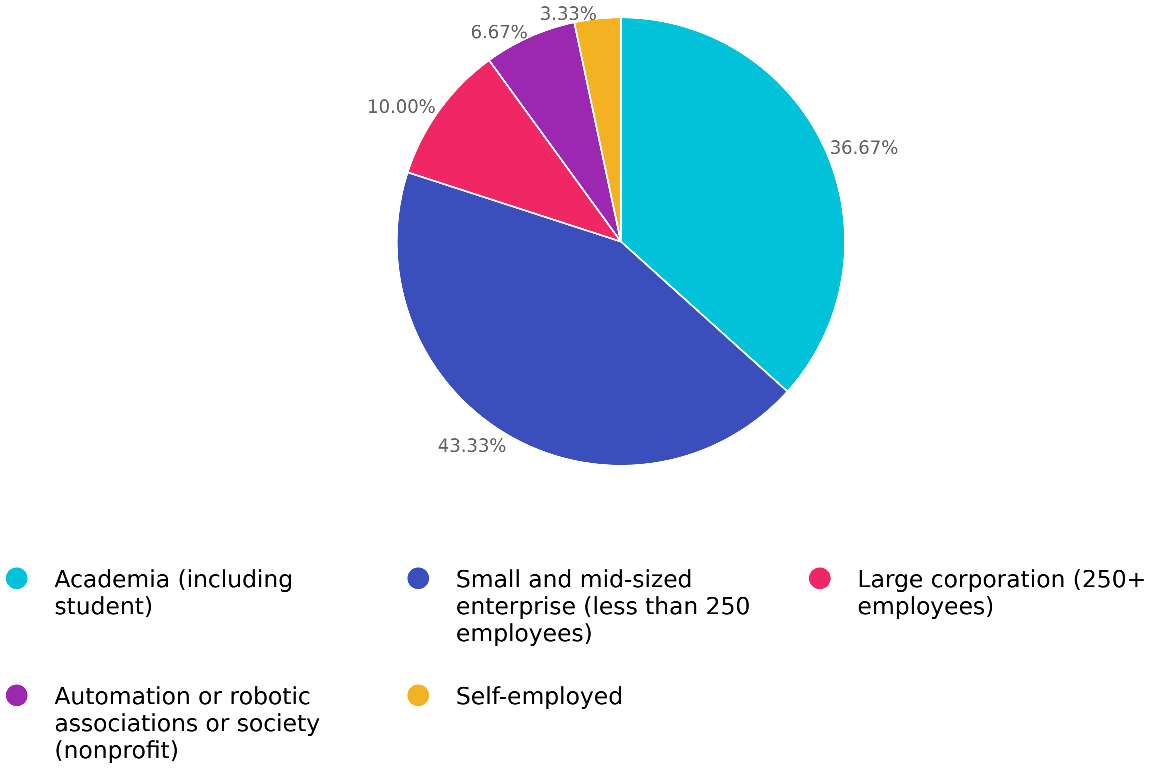
Page 1

Your Role

Q1

What entity do you work for?

Answered: 60   Skipped: 0

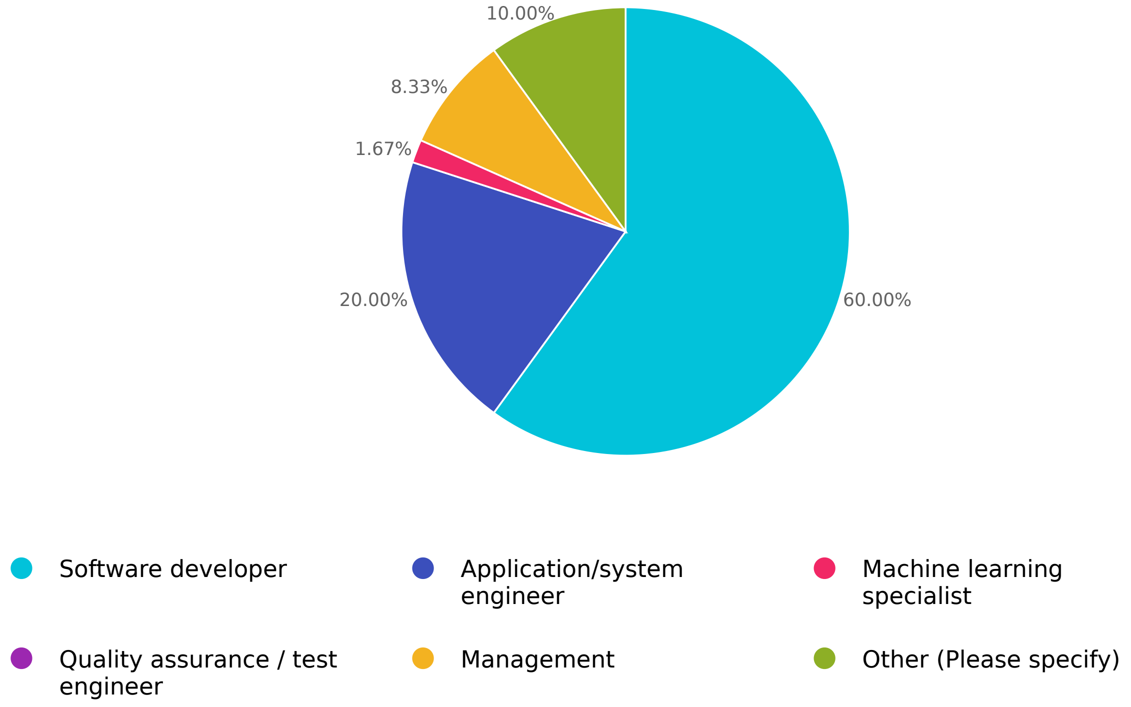


Choices	Response percent	Response count
Academia (including student)	36.67%	22
Small and mid-sized enterprise (less than 250 employees)	43.33%	26
Large corporation (250+ employees)	10.00%	6
Automation or robotic associations or society (nonprofit)	6.67%	4
Self-employed	3.33%	2

Q2

Please select the option that best describes your role.

Answered: 60 Skipped: 0



Choices	Response percent	Response count
Software developer	60.00%	36
Application/system engineer	20.00%	12
Machine learning specialist	1.67%	1
Quality assurance / test engineer	0.00%	0
Management	8.33%	5
Other (Please specify)	10.00%	6

Other (Please specify)

1. Academic
2. Researcher
3. Phd student
4. Researcher
5. Systems and Control engineer
6. Electronic and Robotics Engineer

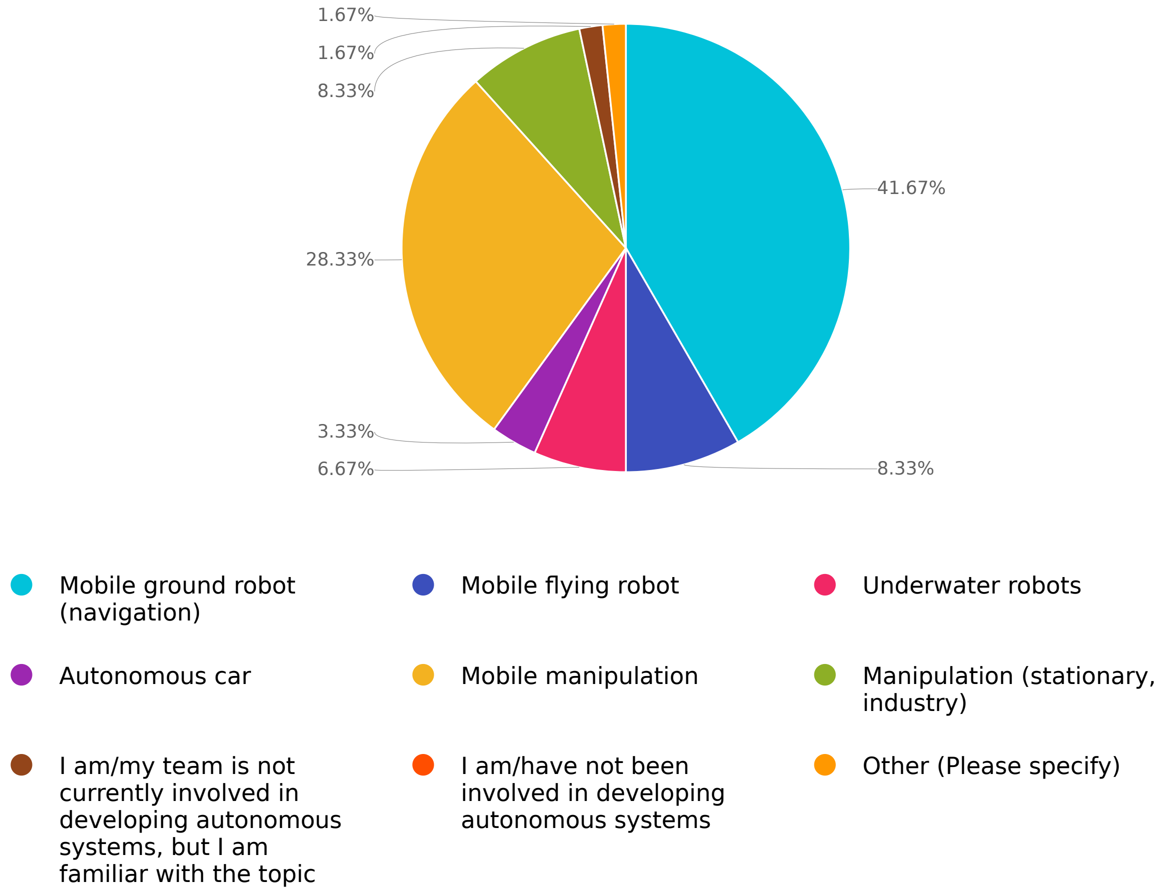
Page 2

Your Role

Q3

Which of the following best describes the autonomous system you are mainly developing?

Answered: 60 Skipped: 0



Choices	Response percent	Response count
Mobile ground robot (navigation)	41.67%	25
Mobile flying robot	8.33%	5
Underwater robots	6.67%	4
Autonomous car	3.33%	2
Mobile manipulation	28.33%	17
Manipulation (stationary, industry)	8.33%	5
I am/my team is not currently involved in developing autonomous systems, but I am familiar with the topic	1.67%	1
I am/have not been involved in developing autonomous systems	0.00%	0
Other (Please specify)	1.67%	1

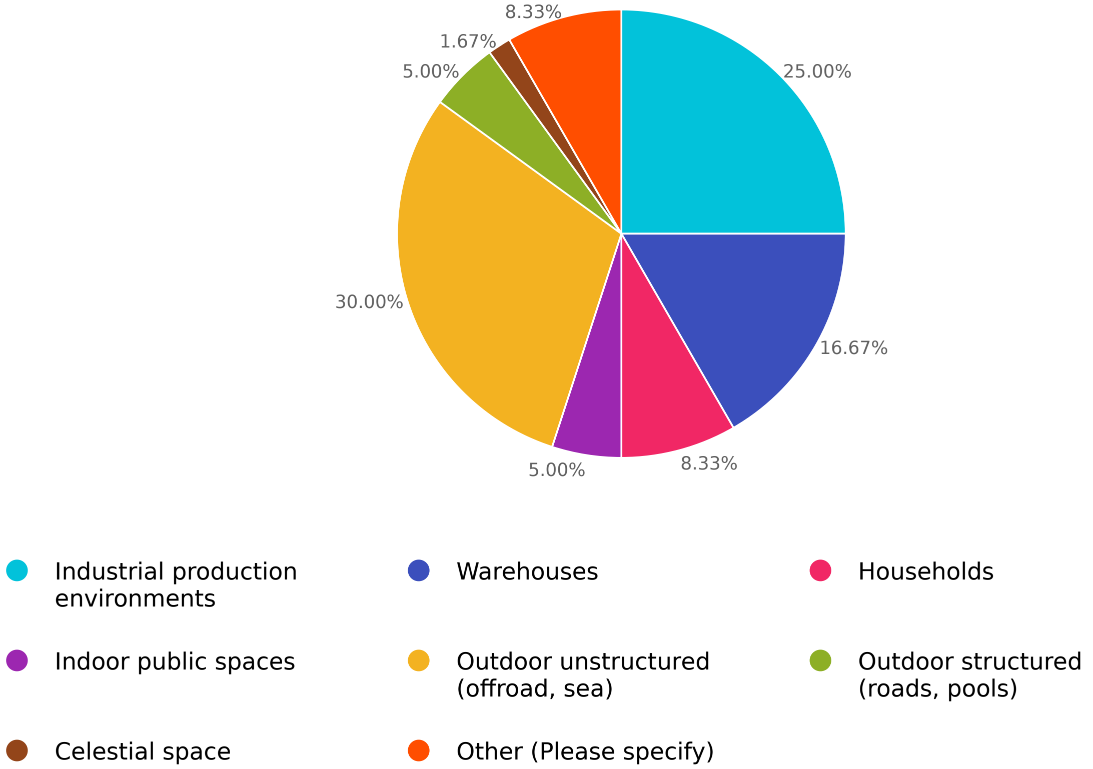
Other (Please specify)

1. Space robot

Q4

In which environments do the autonomous systems you are mainly developing typically operate?

Answered: 60 Skipped: 0



Choices	Response percent	Response count
Industrial production environments	25.00%	15
Warehouses	16.67%	10
Households	8.33%	5
Indoor public spaces	5.00%	3
Outdoor unstructured (offroad, sea)	30.00%	18
Outdoor structured (roads, pools)	5.00%	3
Celestial space	1.67%	1
Other (Please specify)	8.33%	5

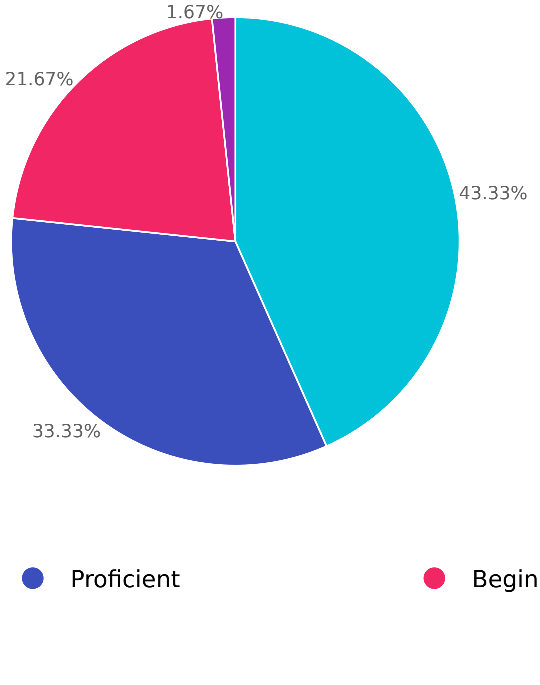
Other (Please specify)

1. Mines
2. Environment-agnostic, we make software tools
3. Offshore
4. Hospitals
5. Industrial and household

Q5

What is your level of understanding in the area of deliberation for autonomous systems (e.g. situation modelling, task planning, skills/capabilities/behavior definition)?

Answered: 60 Skipped: 0



Choices	Response percent	Response count
Expert	43.33%	26
Proficient	33.33%	20
Beginner	21.67%	13
Not competent / no knowledge	1.67%	1

Deliberation Language

Terminology:

Deliberation language: The language used to define a deliberation logic (e.g. BT, FSM, MC, ...)

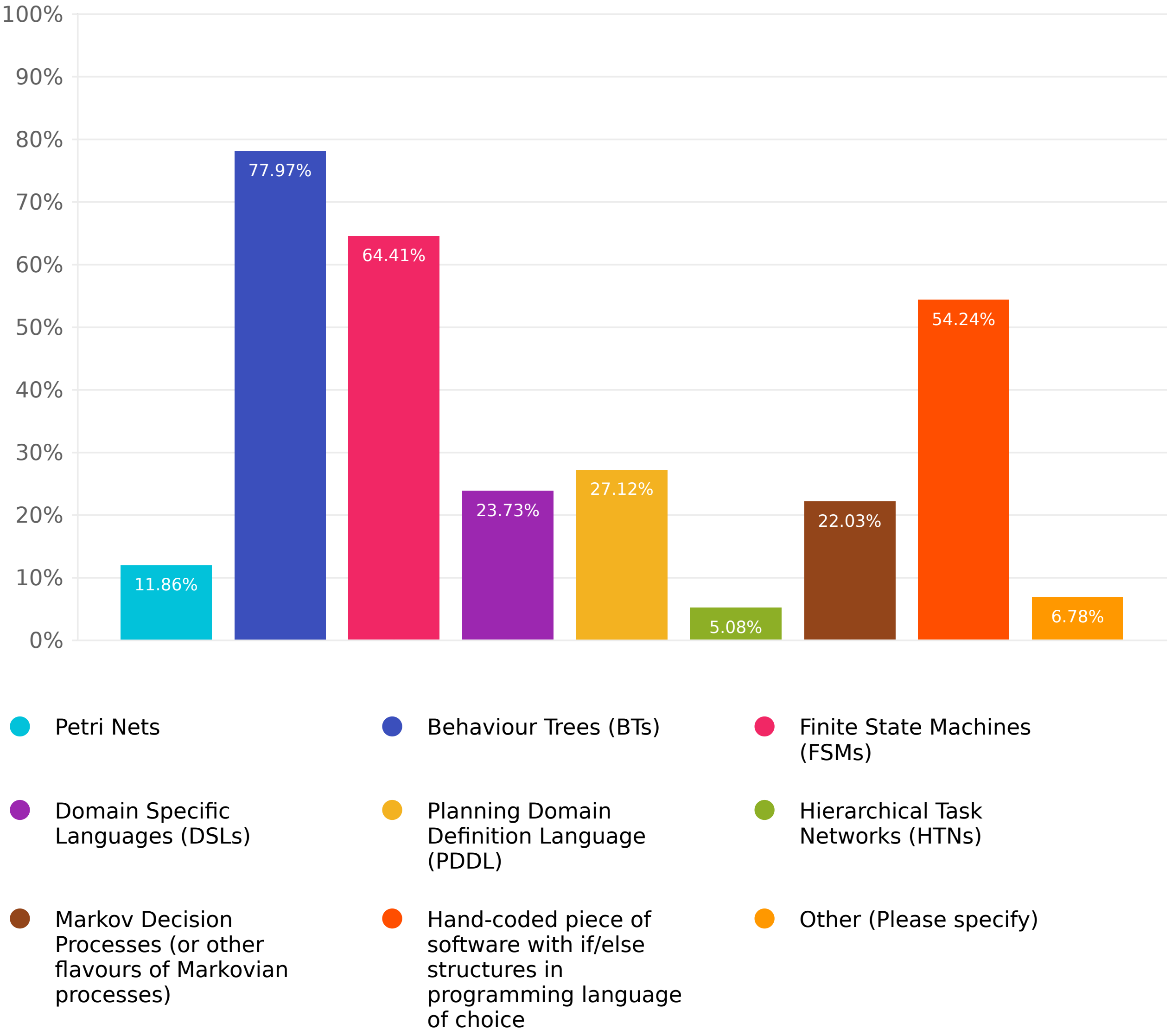
Deliberation logic: The specific instance of the logic the system will execute (e.g. Point-to-point navigation)

Deliberation engine: The actual SW/library that runs a specific deliberation language (e.g. SMACC, FlexBe, BehaviorTree.CPP, ...)

Q6

Which of the following deliberation languages did you use regularly during the past year? Please select all that apply.

Answered: 59 Skipped: 1



Choices	Response percent	Response count
Petri Nets	11.86%	7
Behaviour Trees (BTs)	77.97%	46
Finite State Machines (FSMs)	64.41%	38
Domain Specific Languages (DSLs)	23.73%	14
Planning Domain Definition Language (PDDL)	27.12%	16
Hierarchical Task Networks (HTNs)	5.08%	3
Markov Decision Processes (or other flavours of Markovian processes)	22.03%	13
Hand-coded piece of software with if/else structures in programming language of choice	54.24%	32
Other (Please specify)	6.78%	4

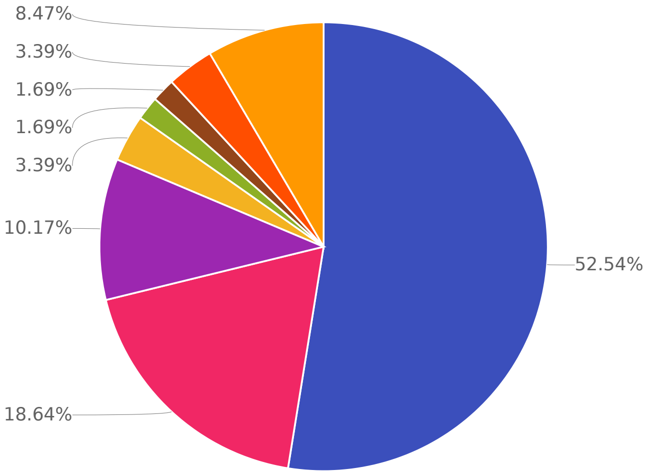
Other (Please specify)

1. Prolog, BDI agents
2. Multiple methodologies
3. Milp based allocation and scheduling
4. HSM

Q7

Which of the following is your *preferred* **deliberation language**?

Answered: 59 Skipped: 1



● Petri Nets	● Behaviour Trees (BTs)	● Finite State Machines (FSMs)
● Domain Specific Languages (DSLs)	● Planning Domain Definition Language (PDDL)	● Hierarchical Task Networks (HTNs)
● Markov Decision Processes (or other flavours of Markovian processes)	● Hand-coded piece of software with if/else structures in programming language of choice	● Other (Please specify)

Choices	Response percent	Response count
Petri Nets	0.00%	0
Behaviour Trees (BTs)	52.54%	31
Finite State Machines (FSMs)	18.64%	11
Domain Specific Languages (DSLs)	10.17%	6
Planning Domain Definition Language (PDDL)	3.39%	2
Hierarchical Task Networks (HTNs)	1.69%	1
Markov Decision Processes (or other flavours of Markovian processes)	1.69%	1
Hand-coded piece of software with if/else structures in programming language of choice	3.39%	2
Other (Please specify)	8.47%	5

Other (Please specify)

1. ROS smach
2. BDI agents with Jason language
3. HSM
4. Hybrid control systems
5. Wrong question: different "languages" serve different purposes!

Q8

Are there any additional strengths and/or limitations of your *preferred* **deliberation language** when developing autonomous systems? Please provide a brief explanation.

Answered: 19 Skipped: 41

1. (Domain Specific Languages (DSLs) being the preferred deliberation language) Must allow to use at different abstraction levels.
2. (Finite State Machines (FSMs) being the preferred deliberation language) Using HFSM (Hierarchical Finite State Machines) has several advantages in my opinion. Easy to map to user-understandable behavior, easy to debug, easy to understand, modify and so on. The "Sub-Machines" can be used to control specific parts/systems of the robot. By limitations, it's not as modular as it should, be since most states cannot be re-used, or need to be implemented with a lot of abstraction to be re-used.
3. (Finite State Machines (FSMs) being the preferred deliberation language) Hard to compose once the behaviors become more and more complex.
4. (Behaviour Trees (BTs) being the preferred deliberation language) Lack of clear documentation and basic IF/ELSE structure
5. (Domain Specific Language DSL being the preferred deliberation language) Using a fully general language like PDDL often has too many features and results in unnecessary complexity. A reasonable DSL tuned towards the expressibility and features needed in a given project is often simpler to reason over and use. Internally we use a task planner that provides similar capabilities to PDDL 2.1 but using a project-specific DSL allows for additional type safety.
6. (Behaviour Trees (BTs) being the preferred deliberation language) Strengths: Mentioned above Limitations: I think the amount of boilerplate code required is high.
7. (Planning Domain Definition Language (PDDL) being the preferred deliberation language) Strength: PDDL (or automated planning with a model, in general) may be challenging to set up and debug, but once it works it can solve a variety of goals under a variety of initial conditions, which makes it quite powerful. Limitation: However, automated task planning still has many practical challenges in its real-world deployments, which is a current limitation of the field but also an exciting area that I work on addressing.
8. (Other-ROS smach being the preferred deliberation language) Easy to visualize. e.g. GUI based.
9. (Hand-coded piece of software with if/else structures in programming language of choice being the preferred deliberation language) Be language agnostic. It really should not matter. Universal is universal.
10. (Domain specific Languages DSL being the preferred deliberation language) Easy to integrate multiple navigation types. Semantic navigation, metric navigation etc
11. (Finite State Machines (FSMs) being the preferred deliberation language) It can get complex if the application scales.
12. (Behaviour Trees (BTs) being the preferred deliberation language) Lack of the BT implementation in most of the AMR API, I have to construct behaviors from the basic commands and logic each time.
13. (Other-HSM being the preferred deliberation language) (Translated) I think there is essentially no difference between HSM and BT, although BT provides configurable capabilities in the form of XML file descriptions. But I think if you want, HSM can also expand this function. In the Nav2 project, to use BT, I need to first understand the concepts of special nodes, such as "FALLBACK". This doesn't seem as easy to understand as a flowchart or HSM's state transition diagram. In the end I think the function of the robot is definite and single. We can use HSM's state transition diagram to express the functional logic very well. This will be easier to understand. But it is undeniable that BT is very popular now, and I will also learn to use it.
14. (Finite State Machines (FSMs) being the preferred deliberation language) I did not really have time to explore other options.
15. (Finite State Machines (FSMs) being the preferred delibreation language) An additional strength is modularity (can be achieved with appropriate development). A limitation, not just of FSMs, but of all other deliberation languages, is that they require a certain level of proficiency to be used effectively; this is particularly a problem when working with less experienced developers, who struggle to understand how to properly use the formalism.
16. (Other-Hybrid control systems being the preferred deliberation language) 100% Formal guarantees is key

17. (Other being the preferred deliberation language) "Preference" is the wrong question to ask. Just like "easy to use". What is important is the semantic richness and completeness that is offered, and the insights about which of the (many!) decision making types/needs are supported.

18. (Finite State Machines (FSMs) being the preferred deliberation language) Domain Specific Languages (DSLs) are a great option for their flexibility and expressiveness, the choice between the others and DSLs depends on the specific requirements of the system being designed. Each method has its strengths and weaknesses, and the best choice would depend on factors such as the complexity of the system, the need for concurrency or synchronization and the level of domain-specific knowledge involved.

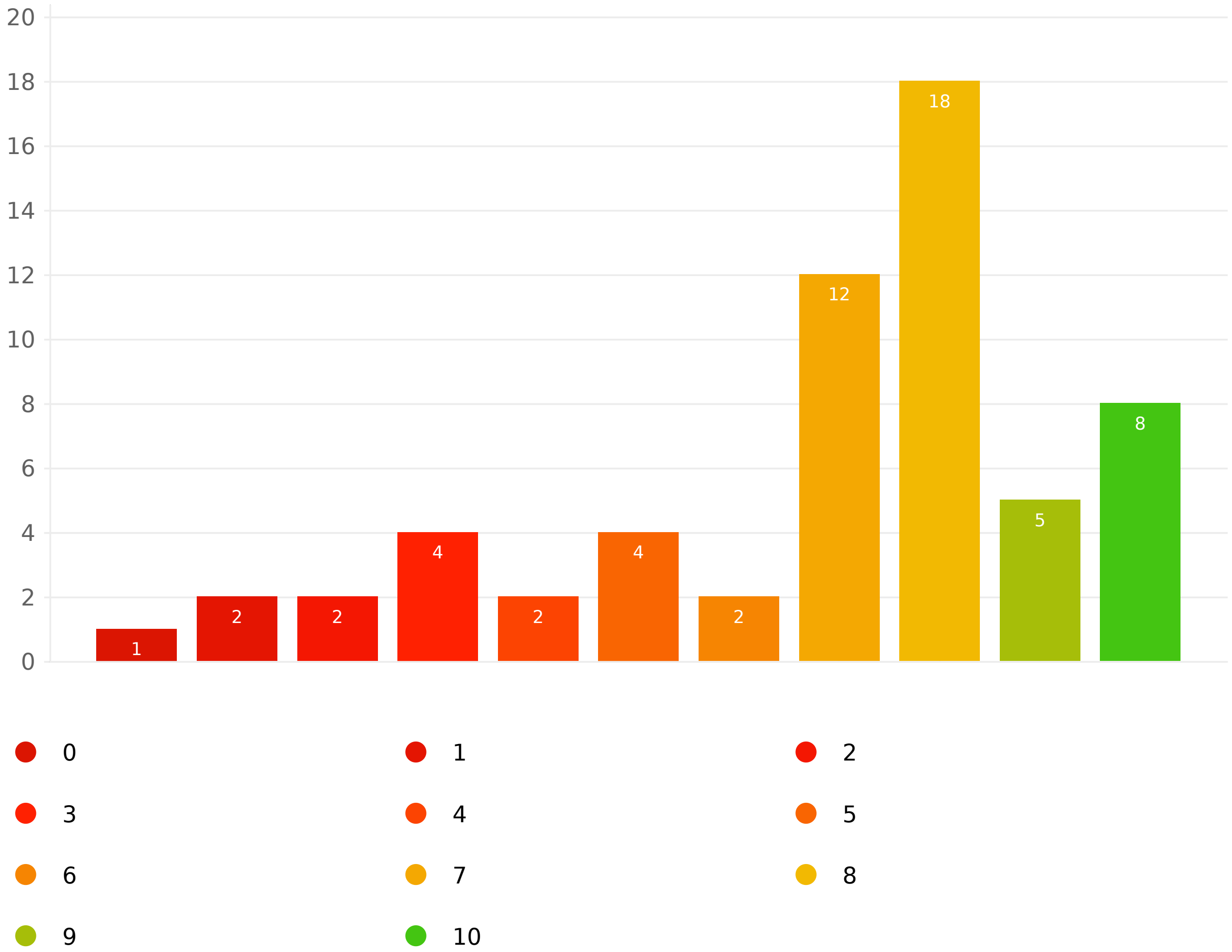
19. (Behaviour Trees (BTs) being the preferred deliberation language) Strengths: Fully scalable and community that has provided quite a lot of material of it. Limitations: Not easy to implement from scratch without some templated, in base of ROS.

Q9

How keen are you on using Behavior Trees when developing an autonomous system?

0-10 rating scale
0 = Not very keen
10 = Very keen

Answered: 60 Skipped: 0



Choices	Response percent	Response count
0 (Not very keen)	1.67%	1
1	3.33%	2
2	3.33%	2
3	6.67%	4
4	3.33%	2
5	6.67%	4
6	3.33%	2
7	20.00%	12
8	30.00%	18
9	8.33%	5
10 (Very keen)	13.33%	8

Deliberation Logic and Engine

Terminology:

Deliberation language: The language used to define a deliberation logic (e.g. BT, FSM, MC, ...)

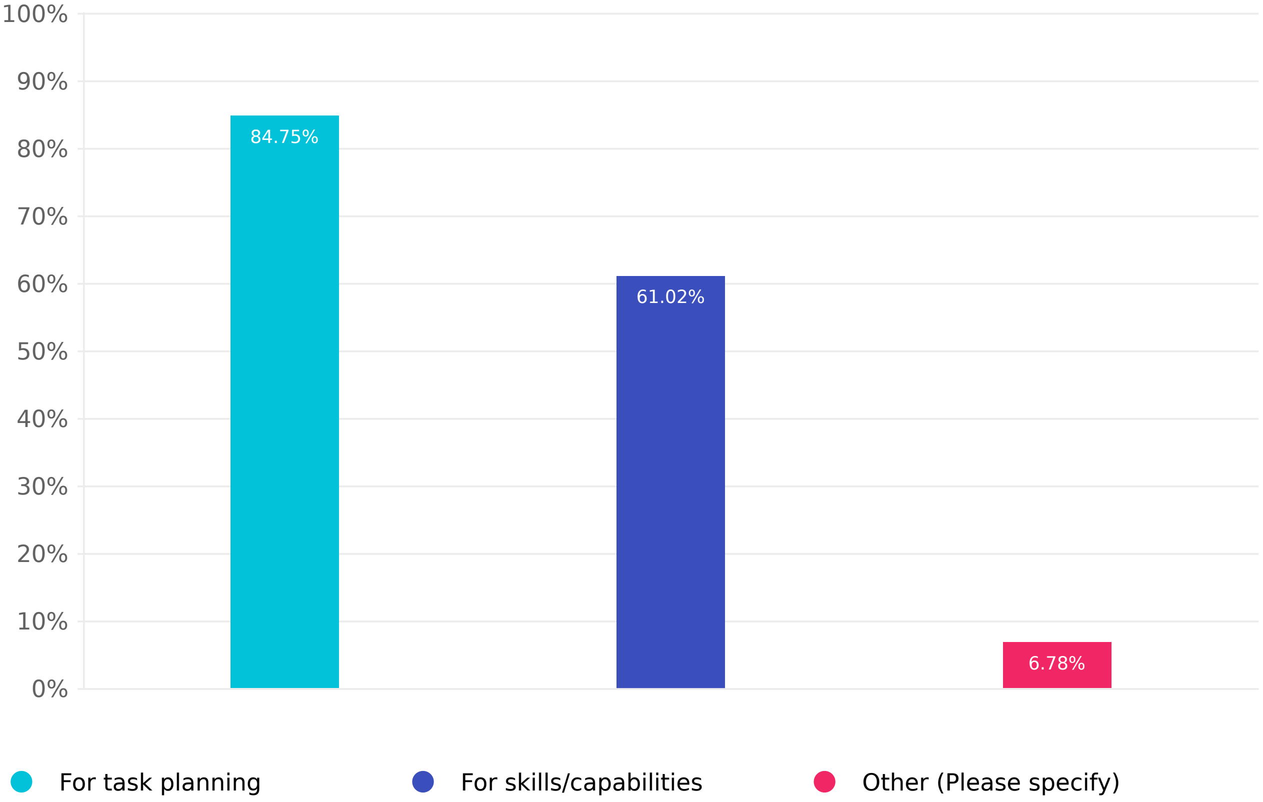
Deliberation logic: The specific instance of the logic the system will execute (e.g. Point-to-point navigation)

Deliberation engine: The actual SW/library that runs a specific deliberation language (e.g. SMACC, FlexBe, BehaviorTree.CPP, ...)

Q10

For which specific features of your autonomous system do you use deliberation? Please select all that apply.

Answered: 59 Skipped: 1



Choices	Response percent	Response count
For task planning	84.75%	50
For skills/capabilities	61.02%	36
Other (Please specify)	6.78%	4

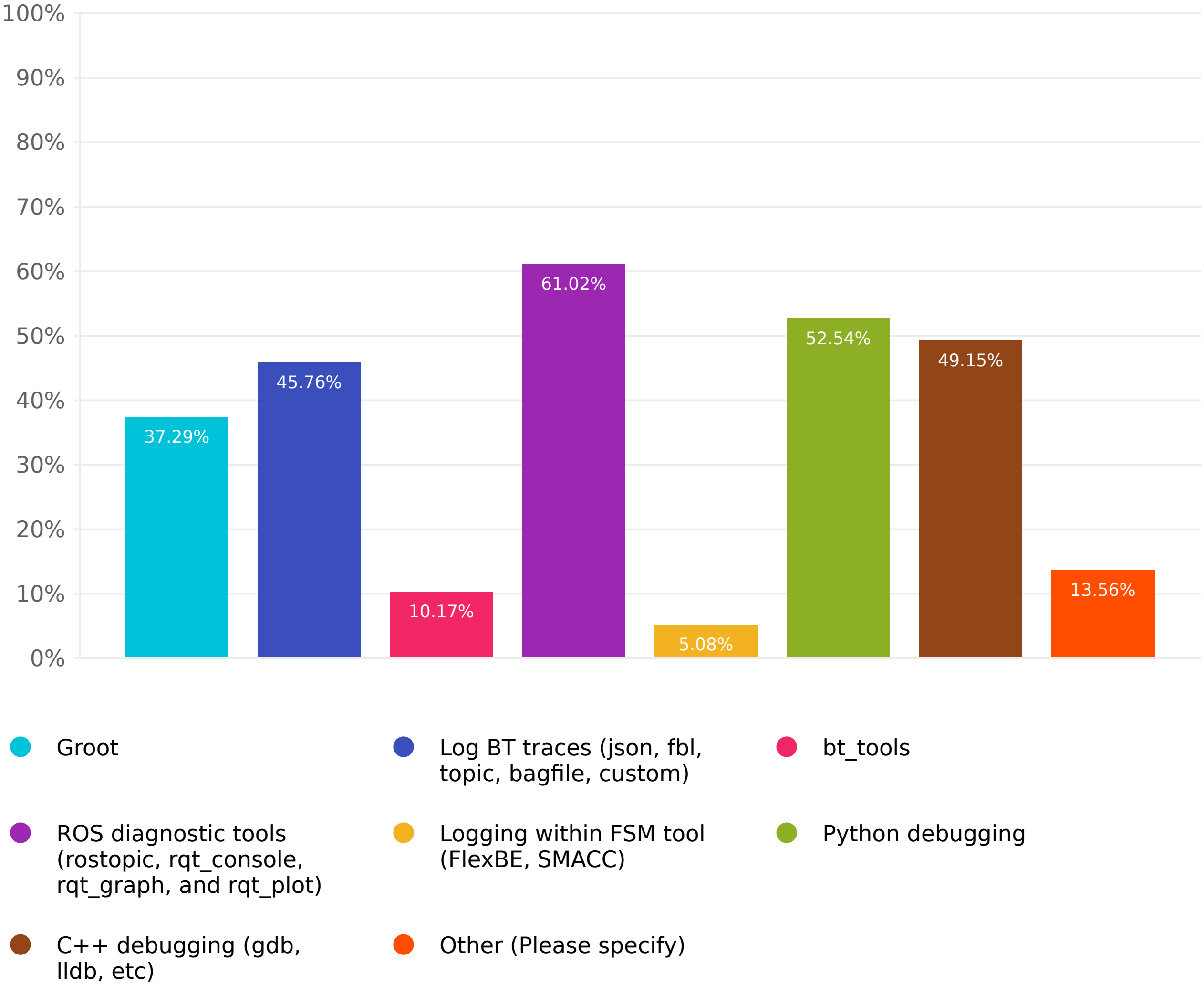
Other (Please specify)

1. Navigation
2. For configuring the architecture of the robot
3. Fallback logic and recovery
4. Navigation/Behavior selection

Q11

What debugging methods or tools do you use to introspect your **deliberation logic** executions? Please select all that apply.

Answered: 59 Skipped: 1



Choices	Response percent	Response count
Groot	37.29%	22
Log BT traces (json, fbl, topic, bagfile, custom)	45.76%	27
bt_tools	10.17%	6
ROS diagnostic tools (rostopic, rqt_console, rqt_graph, and rqt_plot)	61.02%	36
Logging within FSM tool (FlexBE, SMACC)	5.08%	3
Python debugging	52.54%	31
C++ debugging (gdb, lldb, etc)	49.15%	29
Other (Please specify)	13.56%	8

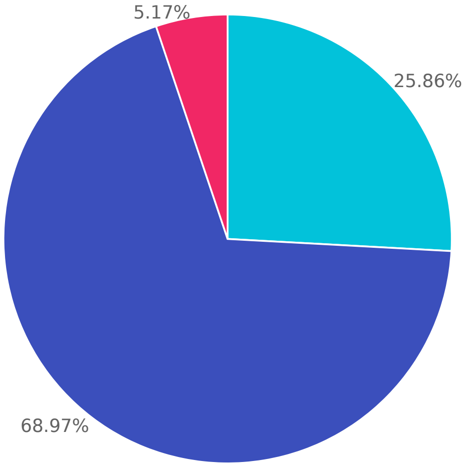
Other (Please specify)

1. Not specified
2. Our task planner provides introspection/"explain" logging
3. MoveIt Studio
4. Custom-based
5. Whatever works best for the issue at hand.
6. Internal BT implementation
7. SimPy implementation suited to BT execution
8. log, print

Q12

How do you ensure your **deliberation logic** can deal with events that are not explicitly foreseen?

Answered: 58 Skipped: 2



● Add reactions blocking the deliberation engine execution (outside deliberation engine)

● Define generic fallbacks in the deliberation logic

● Other (Please specify)

Choices	Response percent	Response count
Add reactions blocking the deliberation engine execution (outside deliberation engine)	25.86%	15
Define generic fallbacks in the deliberation logic	68.97%	40
Other (Please specify)	5.17%	3

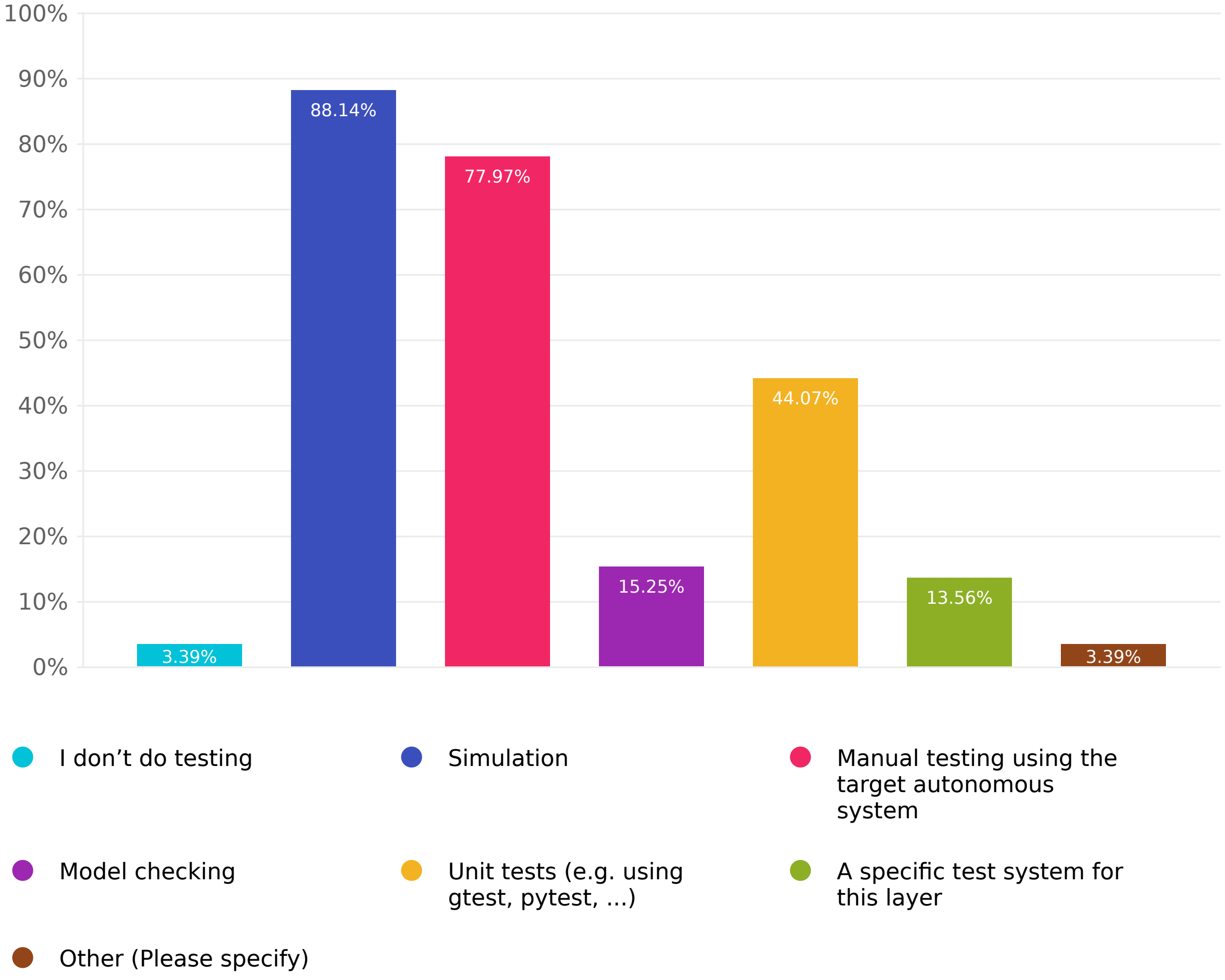
Other (Please specify)

1. We run our task planner in a closed-loop manner after every skill execution
2. DSL with design alternatives to runtime reconfiguration
3. Continual learning

Q13

How do you typically test your **deliberation logic**? Please select all that apply.

Answered: 59   Skipped: 1



Choices	Response percent	Response count
I don’t do testing	3.39%	2
Simulation	88.14%	52
Manual testing using the target autonomous system	77.97%	46
Model checking	15.25%	9
Unit tests (e.g. using gtest, pytest, ...)	44.07%	26
A specific test system for this layer	13.56%	8
Other (Please specify)	3.39%	2

Other (Please specify)

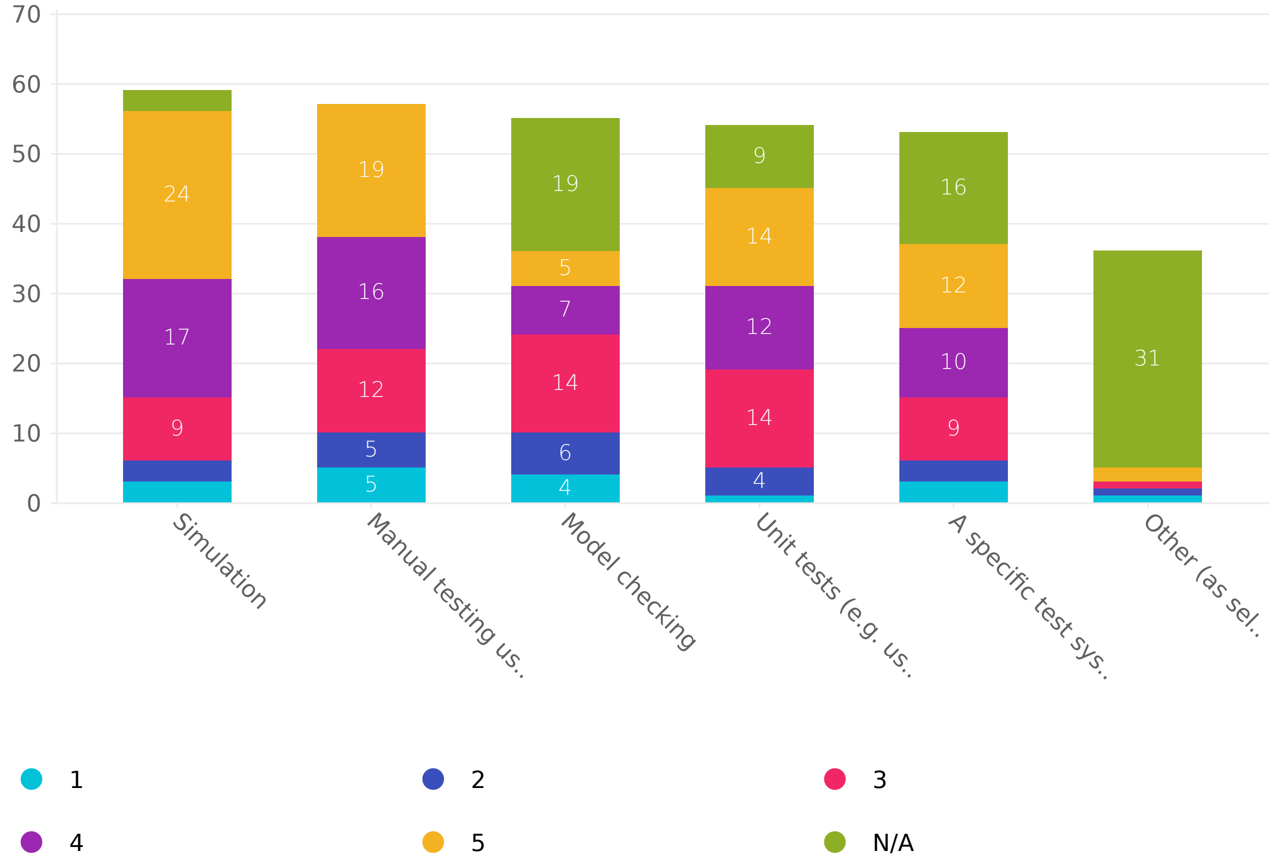
1. Automatic CI pipelines with gazebo simulations
2. log

Q14

Please rate each of the following methods for testing **deliberation logics**.

1-5 rating scale
1 = Very Unsatisfied
5 = Very Satisfied
N/A = Not applicable

Answered: 59 Skipped: 1



Row	1	2	3	4	5	N/A	Average rating	Response count
Simulation	5.08% (3)	5.08% (3)	15.25% (9)	28.81% (17)	40.68% (24)	5.08% (3)	4.00	59
Manual testing using the target autonomous system	8.77% (5)	8.77% (5)	21.05% (12)	28.07% (16)	33.33% (19)	0.00% (0)	3.68	57
Model checking	7.27% (4)	10.91% (6)	25.45% (14)	12.73% (7)	9.09% (5)	34.55% (19)	3.08	55
Unit tests (e.g. using gtest, pytest, ...)	1.85% (1)	7.41% (4)	25.93% (14)	22.22% (12)	25.93% (14)	16.67% (9)	3.76	54
A specific test system for this layer	5.66% (3)	5.66% (3)	16.98% (9)	18.87% (10)	22.64% (12)	30.19% (16)	3.68	53
Other (as selected in previous question)	2.78% (1)	2.78% (1)	2.78% (1)	0.00% (0)	5.56% (2)	86.11% (31)	3.20	36

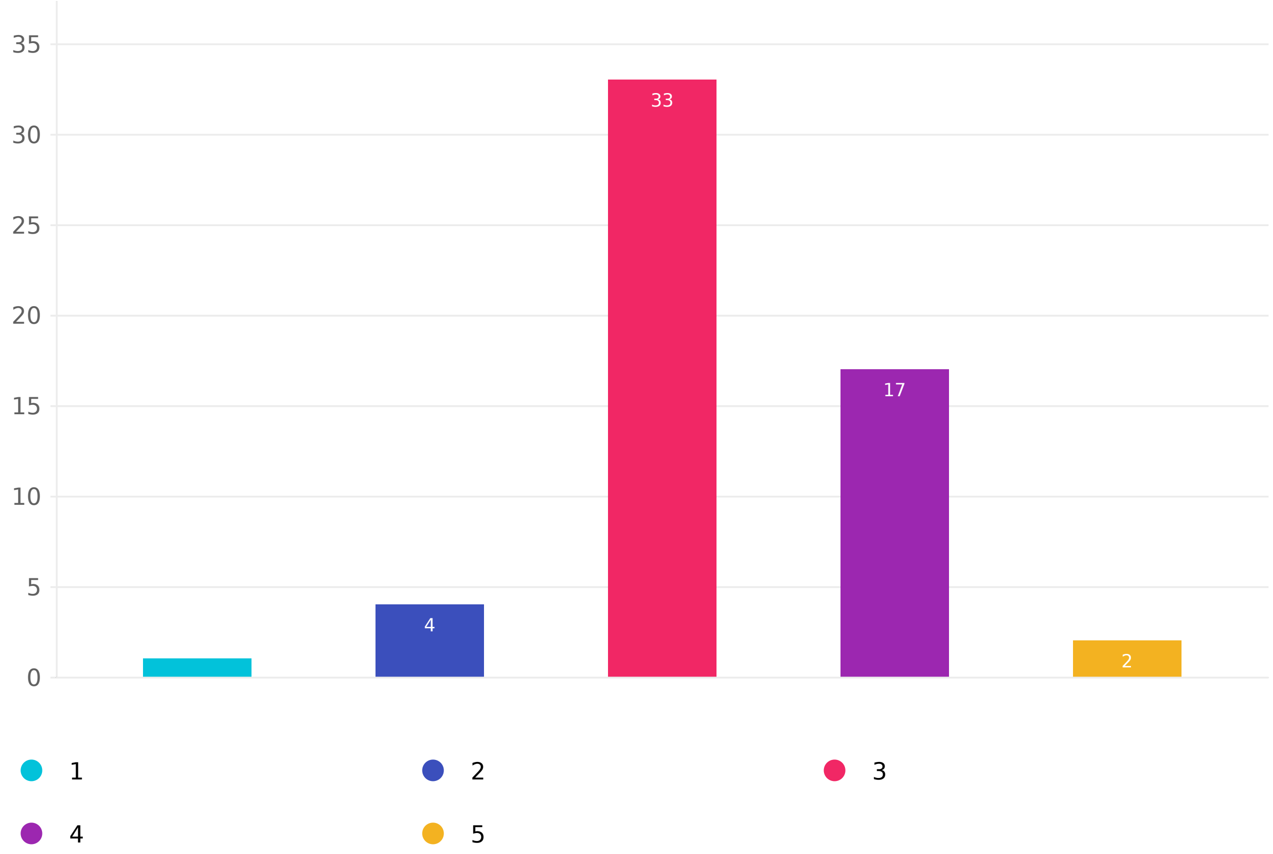
Average rating: 3.67

Q15

How satisfied are you with the current testing methods available for your preferred **deliberation engine**?

1-5 rating scale
1 = Very Unsatisfied
5 = Very Satisfied

Answered: 57 Skipped: 3



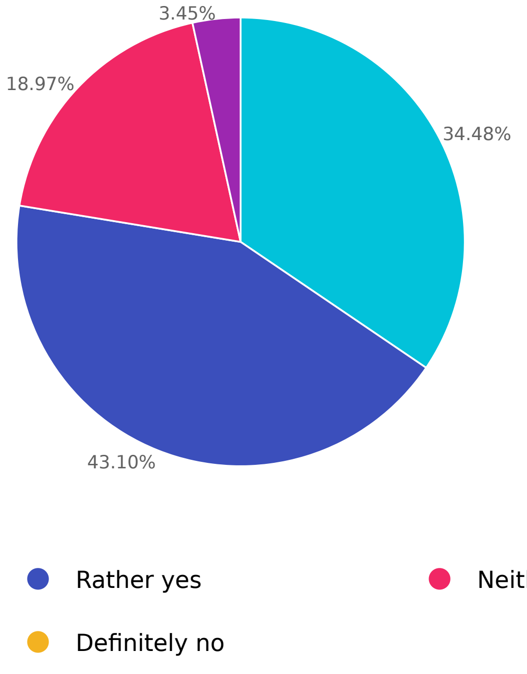
Rating	Response percent	Response count
😠 (Very unsatisfied)	1.75%	1
☹	7.02%	4
😑	57.89%	33
😀	29.82%	17
😍 (Very satisfied)	3.51%	2

Average rating: 3.26

Q16

Do you feel the need for a more systematic testing/verification approach within your **deliberation engine**?

Answered: 58 Skipped: 2



Choices	Response percent	Response count
Definitely yes	34.48%	20
Rather yes	43.10%	25
Neither yes nor no	18.97%	11
Rather no	3.45%	2
Definitely no	0.00%	0
Why? Please provide brief explanation of your choice.		9

Why? Please provide brief explanation of your choice.

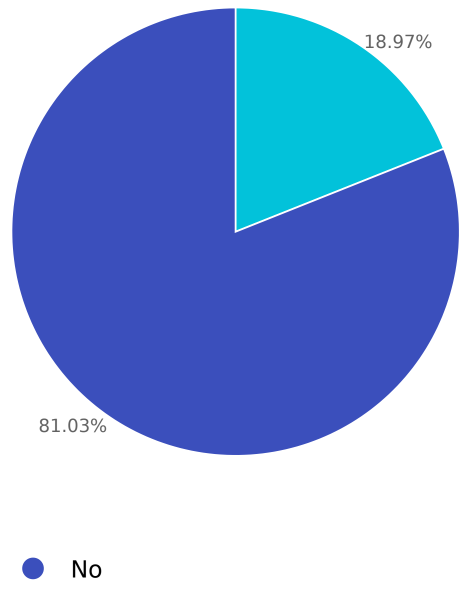
1. (Rather No) System is so complex that high level system tests on the real system appeared to be the most effective (ratio of information obtained vs. time to setup and run the test. In a project where the codebase evolve quickly, we can't afford lots of time re-writting tests endlessly)
2. (Rather Yes) If you allow arbitrarily complex outcomes to occur at runtime, it is very hard to have complete test coverage of all possible states and/or sequences of states. Enumerating all possible states for brute-force testing can be impossible if continuous elements are part of the state.

3. (Definitely Yes) One big challenge is that unit testing is powerful, but especially for complex systems that interface with a communications/hardware layer, it's very difficult to abstract away or mock all the necessary pieces to thoroughly test behaviors.
4. (Definitely Yes) Makes system more robust.
5. (Neither Yes nor No) We found a good way of testing our plans
6. (Definitely Yes) There is no good approach to generic tests of the BT built with the behaviors (using py_trees)
7. (Rather No) because i mean, robot deliberation task is combination of multi simple task. Every simple task test is easy pass to add log. At least for me.
8. (Definitely Yes) _The_ big missing part is to bring in the _context_ of the decision making more explicitly. This can never(?) be done in an "easy to use" way... So, any progress in making verification more systematic is a gain.
9. (Neither yes nor no) The existing approaches seem to provide enough tools for testing such systems, cannot think of how can this be improved.

Q17

Do you already have a model of your **deliberation logic** in a formal representation (e.g. Markovian Processes, Petri Nets, Channel Systems etc.)?

Answered: 58 Skipped: 2



Choices	Response percent	Response count
Yes	18.97%	11
No	81.03%	47

Q18

If Yes, in which format/language?

Answered: 9 Skipped: 51

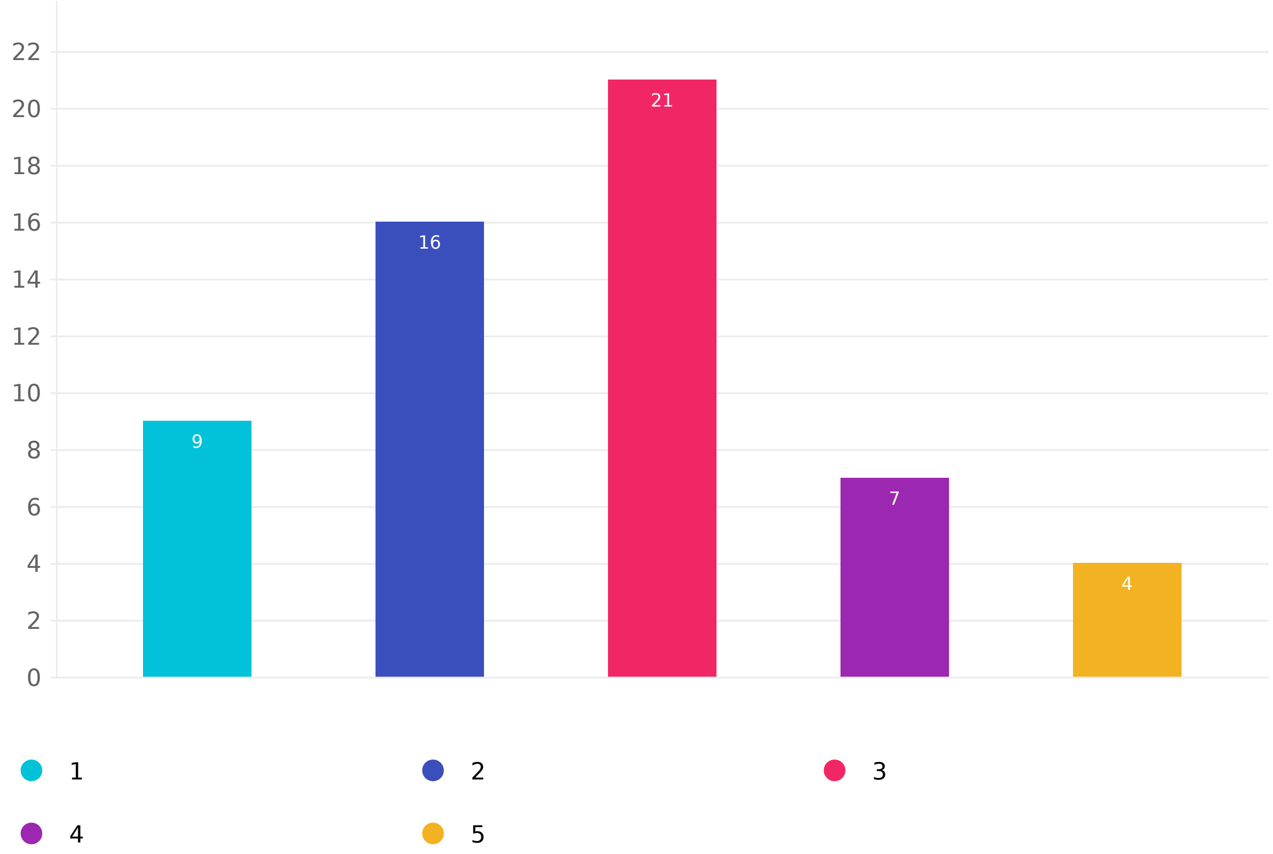
1. Temporal Logic
2. PDDL (or PDDLStream) for task planning, which generates to Behavior Trees for execution.
3. Behavior trees
4. OK, so no general comment option. So putting it here. If you don't know what your focus is and what is wanted - you likely shouldn't be doing this. You started CONVINCE for a reason. What's that reason? I STRONGLY hope you don't put much weight on this survey. You will only get responses from people that fill it out. Which will be a lopsided evaluation of what is needed - at best! Go on your own instinct and base it on the original reasons you started this task. The crowd should never be thought of as an authority. And this 'crowd' is too small to put much value in. Make something GREAT FFS and not just more of what's already done somewhere else. Be general in your solution. Augment and improve. See the big picture. Good luck!
5. BT
6. Milp, mixer integerrimo linear programming as allocation and scheduling model
7. bt xml
8. Markov decision process
9. Domain specific "language"

Q19

Would you in principle spend effort on writing a formal model of your **deliberation logic** if that provides more systematic testing/verification?

1-5 rating scale
1 = Definitely Yes
5 = Definitely No

Answered: 57 Skipped: 3



Rating	Response percent	Response count
1 (Definitely Yes)	15.79%	9
2	28.07%	16
3	36.84%	21
4	12.28%	7
5 (Definitely No)	7.02%	4

Average rating: 2.67

Q20

If Yes, which format/language do you prefer?

Answered: 11 Skipped: 49

1. (Rather Yes) Rust
2. (Rather Yes) Behavior Trees
3. (Definitely Yes) Whatever works best for the current task.
4. (Definitely Yes) Behaviour Trees
5. (Rather Yes) Lean 4, BT, Petri Networks, FSM
6. (Definitely Yes) Python
7. (Rather Yes) Python/C++/Rust
8. (Definitely Yes) Python
9. (Definitely Yes) XML
10. (Rather Yes) Python
11. (Rather Yes) Would prefer C++ or Python. Not a preference on the format.

Q21

If No, please explain why.

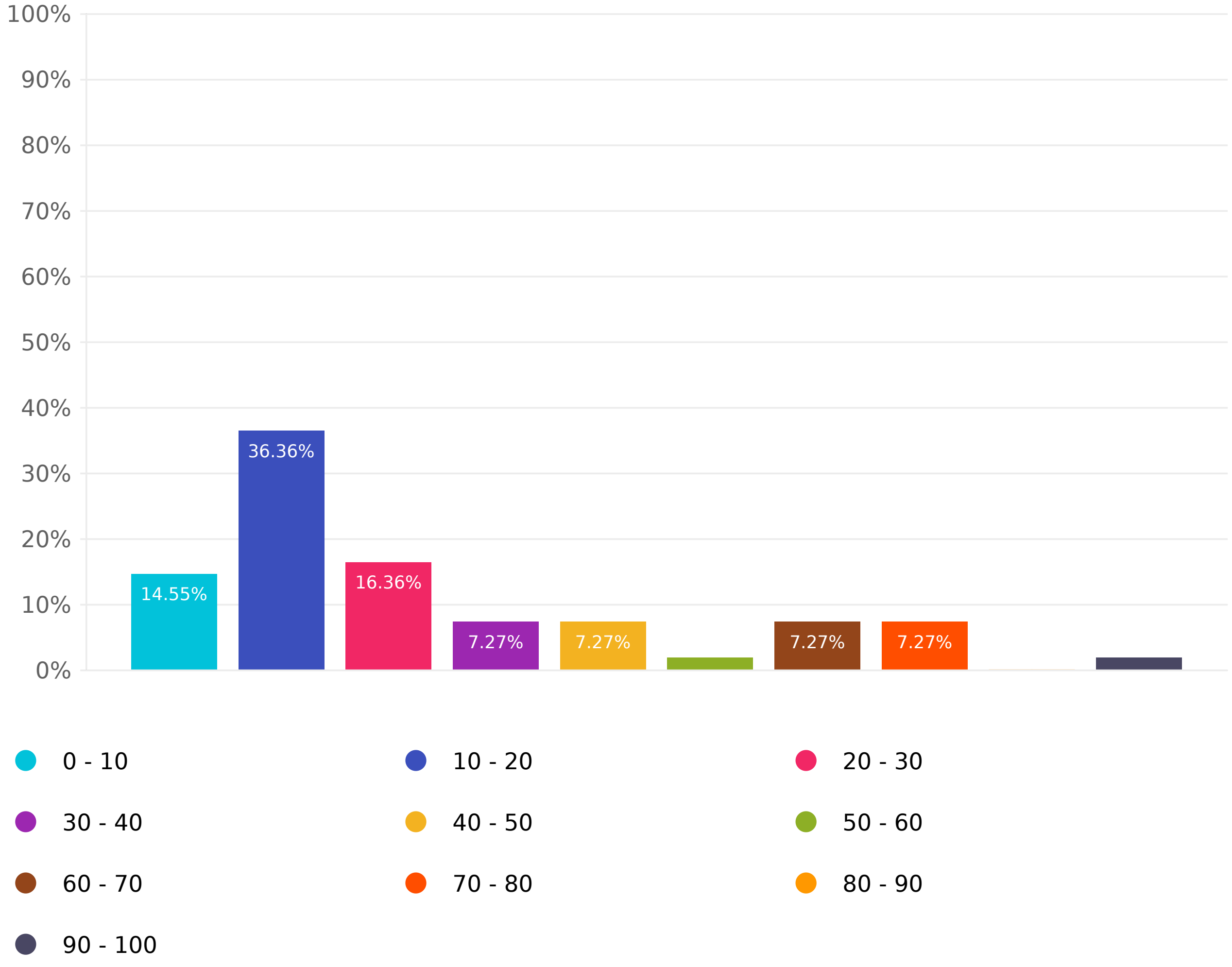
Answered: 5 Skipped: 55

1. (Rather No) We usually do not spend too much time formalizing models since depending of the evolution of the project lot of changes could occur. We have considered it many times, but due to the lack of expert in the field and the complicated learning curve we have always put it off
2. (Definitely No) System is complex but behavior is simple enough to check (navigation)
3. (Rather No) It would be very hard to effectively model our task planner without having to restrict the expressiveness and capabilities.
4. (Rather No) It might be difficult to use, have people understand it and maintain it.
5. (Definitely No) Its a simple logic . We are interested in its practical applications and have intention to scientifically study the logic. The mathematician and logicians have made no efforts to make layman robotisct the importance of petrinets and models in simple words (without mathematics language) the importance and advantages of the modelling. And we are very much happy with testing and validating our results in simulation. Testing code first in simulation and then in hardware gives 100 % confidence that the code is executing as per requirements.

Q22

What effort [in percentage of total development time of your system] are you willing to spend to make systematic verification of your **deliberation logics**?

Answered: 55 Skipped: 5



Choices	Response percent	Response count
0 - 10	14.55%	8
10 - 20	36.36%	20
20 - 30	16.36%	9
30 - 40	7.27%	4
40 - 50	7.27%	4
50 - 60	1.82%	1
60 - 70	7.27%	4
70 - 80	7.27%	4
80 - 90	0.00%	0
90 - 100	1.82%	1

Cross-tab analysis - Preferred deliberation language

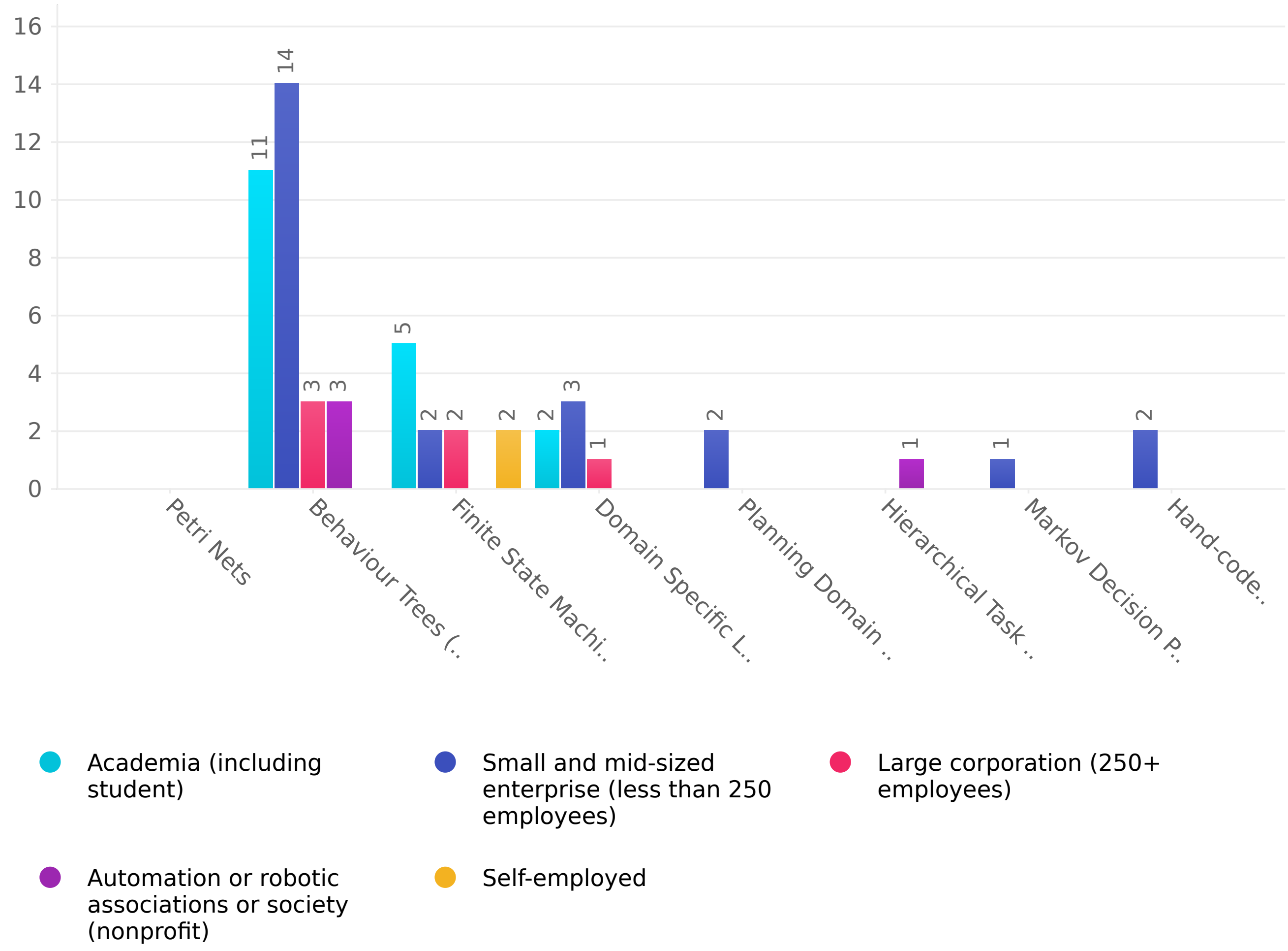
CT1

Which of the following is your *preferred* **deliberation language**?

vs

What entity do you work for?

Answered: 54



CT2

Which of the following is your *preferred* **deliberation language**?

VS

Please provide your opinion of the following statements about your preferred deliberation language when developing autonomous systems.

My preferred deliberation language is...

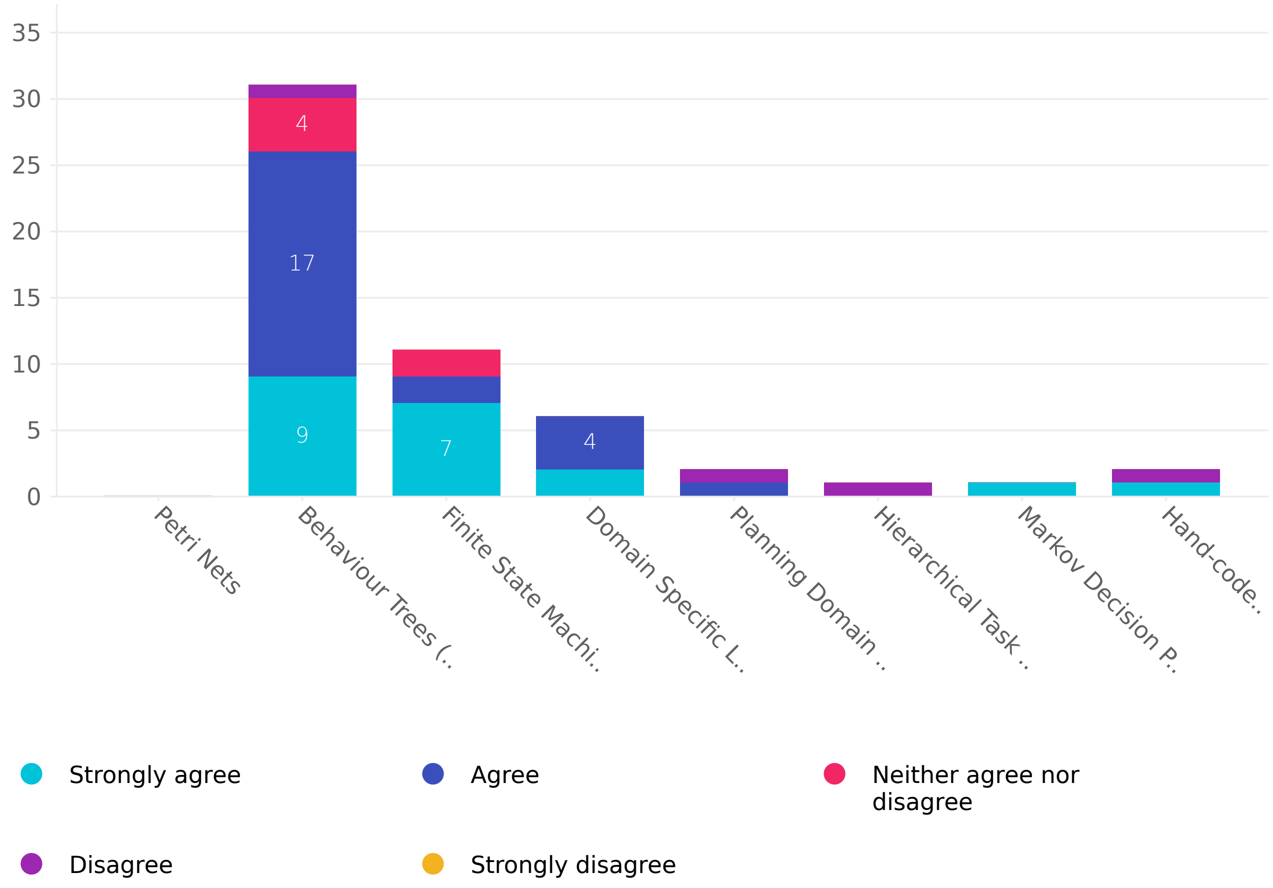
Easy to understand
Easy to implement
Easy to modify and extend
Easy to debug and find mistakes
Easy to compare with another implementation

Answered: 54

Which of the following is your *preferred* **deliberation language**?

VS

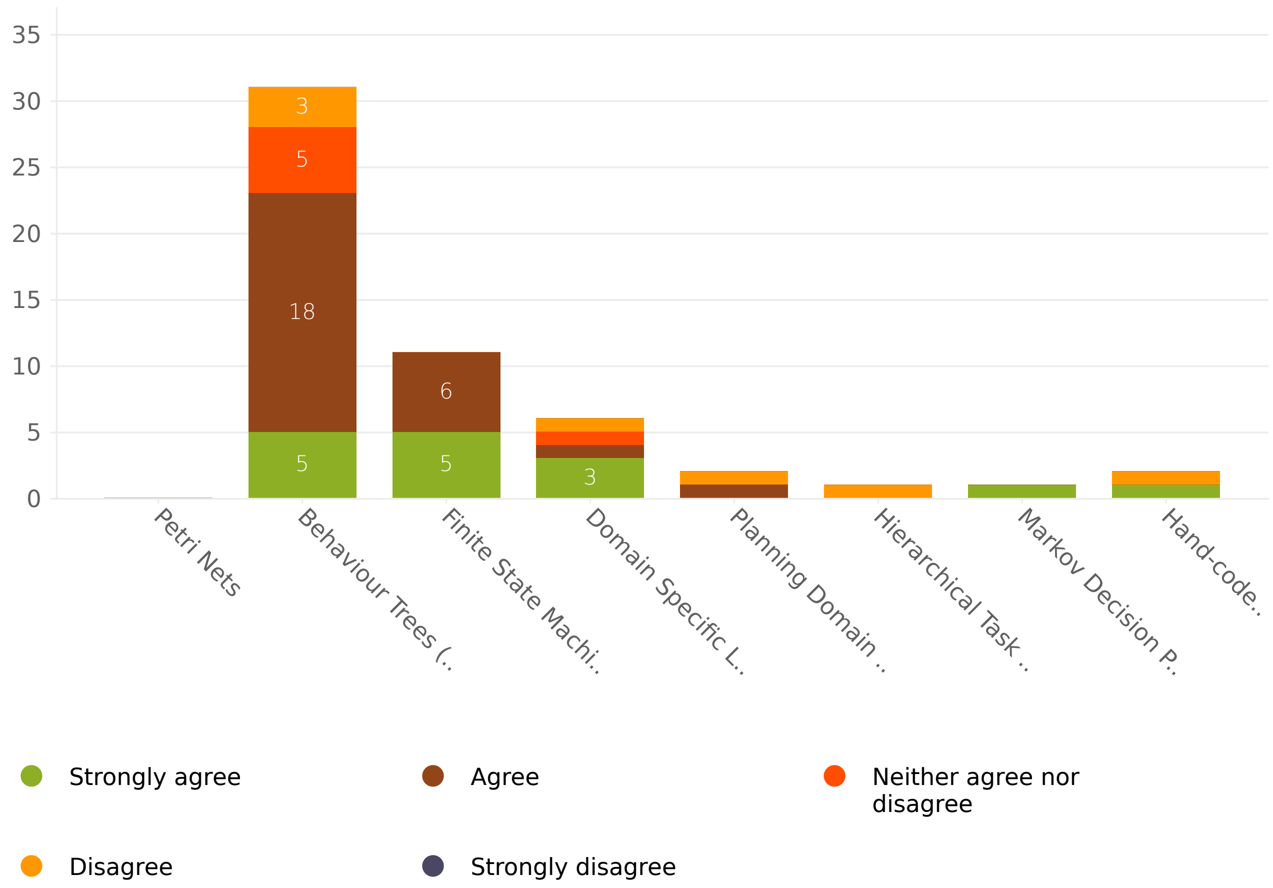
Easy to understand



Row	Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree	Response count
Petri Nets	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0
Behaviour Trees (BTs)	29.03% (9)	54.84% (17)	12.90% (4)	3.23% (1)	0.00% (0)	31
Finite State Machines (FSMs)	63.64% (7)	18.18% (2)	18.18% (2)	0.00% (0)	0.00% (0)	11
Domain Specific Languages (DSLs)	33.33% (2)	66.67% (4)	0.00% (0)	0.00% (0)	0.00% (0)	6
Planning Domain Definition Language (PDDL)	0.00% (0)	50.00% (1)	0.00% (0)	50.00% (1)	0.00% (0)	2
Hierarchical Task Networks (HTNs)	0.00% (0)	0.00% (0)	0.00% (0)	100.00% (1)	0.00% (0)	1
Markov Decision Processes (or other flavours of Markovian processes)	100.00% (1)	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	1
Hand-coded piece of software with if/else structures in programming language of choice	50.00% (1)	0.00% (0)	0.00% (0)	50.00% (1)	0.00% (0)	2

Which of the following is your *preferred* **deliberation language**?

Easy to implement

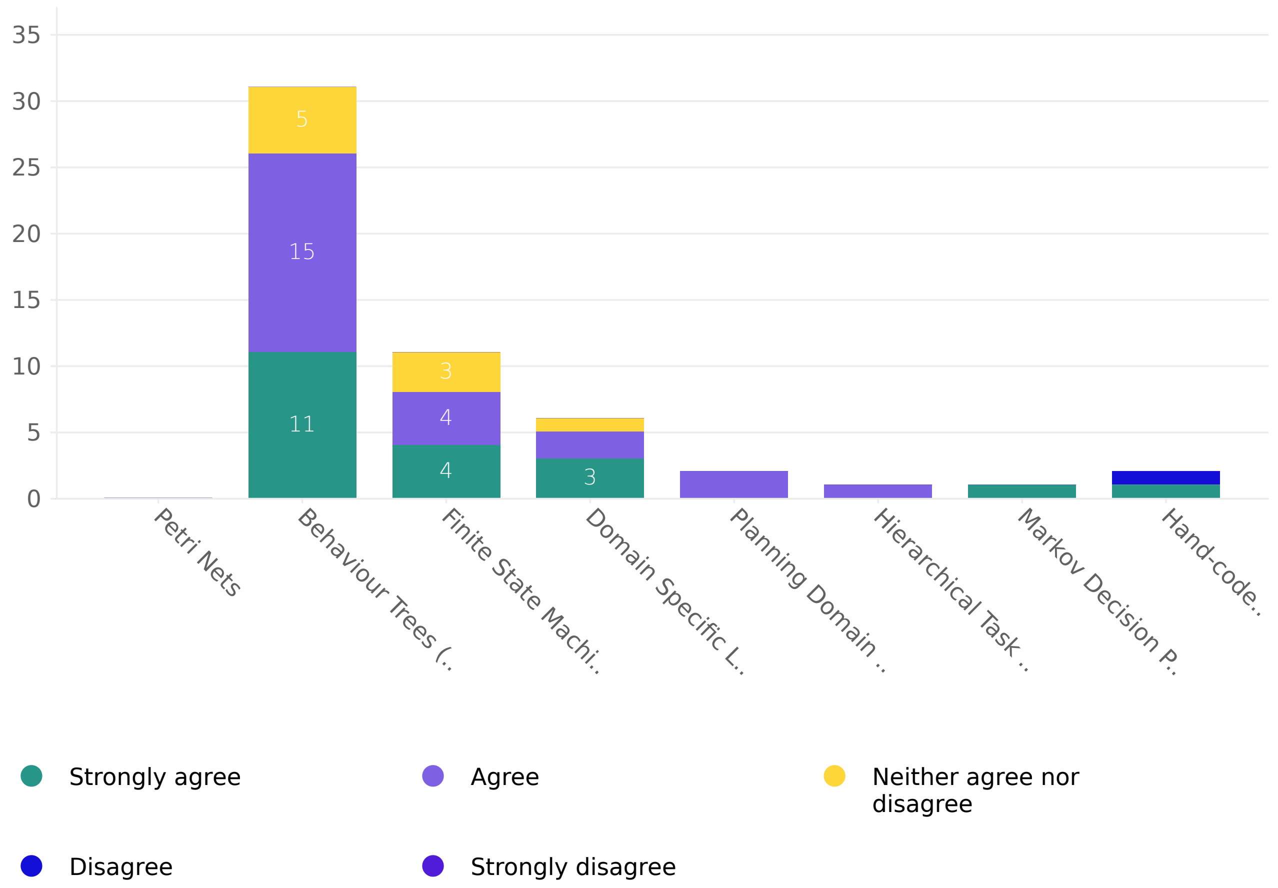


Row	Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree	Response count
Petri Nets	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0
Behaviour Trees (BTs)	16.13% (5)	58.06% (18)	16.13% (5)	9.68% (3)	0.00% (0)	31
Finite State Machines (FSMs)	45.45% (5)	54.55% (6)	0.00% (0)	0.00% (0)	0.00% (0)	11
Domain Specific Languages (DSLs)	50.00% (3)	16.67% (1)	16.67% (1)	16.67% (1)	0.00% (0)	6
Planning Domain Definition Language (PDDL)	0.00% (0)	50.00% (1)	0.00% (0)	50.00% (1)	0.00% (0)	2
Hierarchical Task Networks (HTNs)	0.00% (0)	0.00% (0)	0.00% (0)	100.00% (1)	0.00% (0)	1
Markov Decision Processes (or other flavours of Markovian processes)	100.00% (1)	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	1
Hand-coded piece of software with if/else structures in programming language of choice	50.00% (1)	0.00% (0)	0.00% (0)	50.00% (1)	0.00% (0)	2

VS

Which of the following is your *preferred* **deliberation language**?

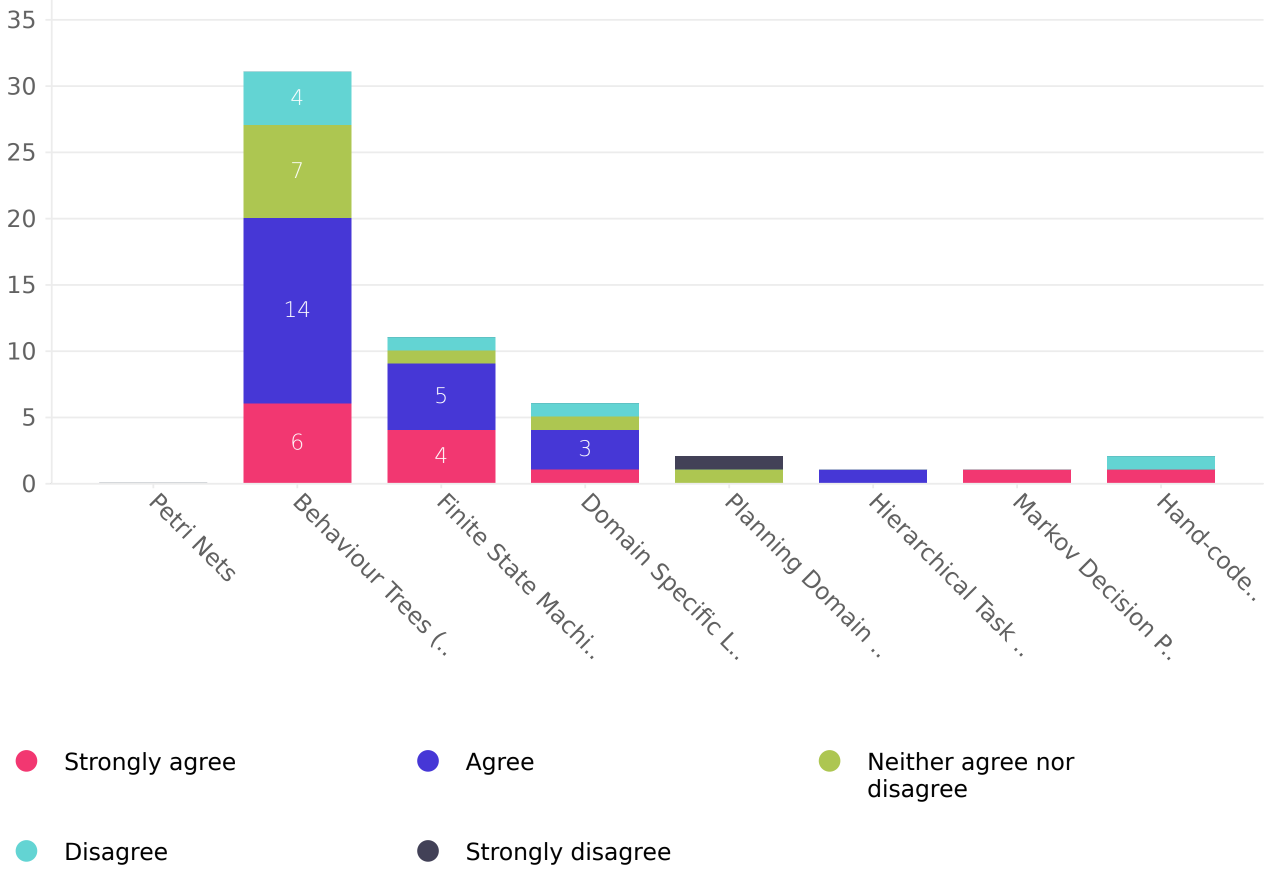
Easy to modify and extend



Row	Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree	Response count
Petri Nets	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0
Behaviour Trees (BTs)	35.48% (11)	48.39% (15)	16.13% (5)	0.00% (0)	0.00% (0)	31
Finite State Machines (FSMs)	36.36% (4)	36.36% (4)	27.27% (3)	0.00% (0)	0.00% (0)	11
Domain Specific Languages (DSLs)	50.00% (3)	33.33% (2)	16.67% (1)	0.00% (0)	0.00% (0)	6
Planning Domain Definition Language (PDDL)	0.00% (0)	100.00% (2)	0.00% (0)	0.00% (0)	0.00% (0)	2
Hierarchical Task Networks (HTNs)	0.00% (0)	100.00% (1)	0.00% (0)	0.00% (0)	0.00% (0)	1
Markov Decision Processes (or other flavours of Markovian processes)	100.00% (1)	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	1
Hand-coded piece of software with if/else structures in programming language of choice	50.00% (1)	0.00% (0)	0.00% (0)	50.00% (1)	0.00% (0)	2

Which of the following is your *preferred* **deliberation language**?

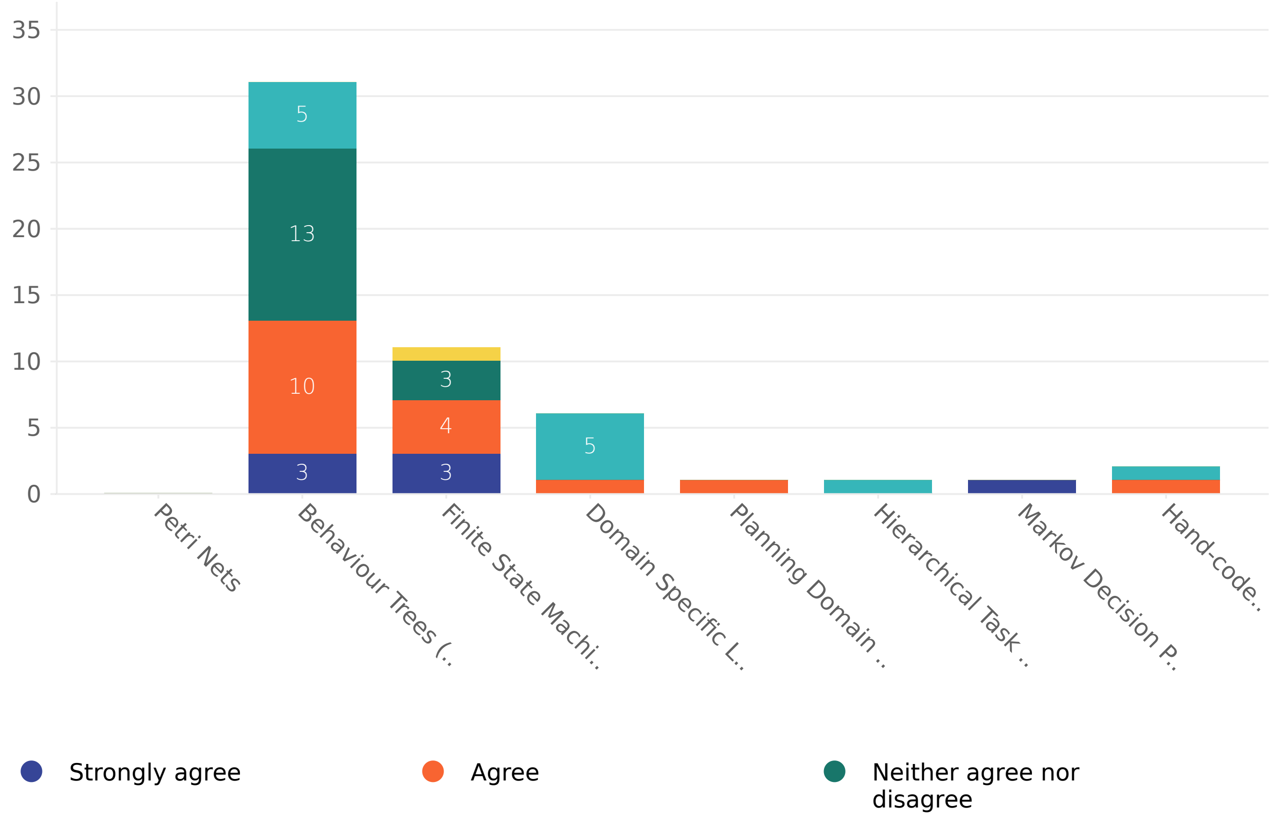
Easy to debug and find mistakes



Row	Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree	Response count
Petri Nets	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0
Behaviour Trees (BTs)	19.35% (6)	45.16% (14)	22.58% (7)	12.90% (4)	0.00% (0)	31
Finite State Machines (FSMs)	36.36% (4)	45.45% (5)	9.09% (1)	9.09% (1)	0.00% (0)	11
Domain Specific Languages (DSLs)	16.67% (1)	50.00% (3)	16.67% (1)	16.67% (1)	0.00% (0)	6
Planning Domain Definition Language (PDDL)	0.00% (0)	0.00% (0)	50.00% (1)	0.00% (0)	50.00% (1)	2
Hierarchical Task Networks (HTNs)	0.00% (0)	100.00% (1)	0.00% (0)	0.00% (0)	0.00% (0)	1
Markov Decision Processes (or other flavours of Markovian processes)	100.00% (1)	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	1
Hand-coded piece of software with if/else structures in programming language of choice	50.00% (1)	0.00% (0)	0.00% (0)	50.00% (1)	0.00% (0)	2

Which of the following is your *preferred* **deliberation language**?

Easy to compare with another implementation



Row	Strongly agree	Agree	Neither agree nor disagree	Disagree	Strongly disagree	Response count
Petri Nets	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0
Behaviour Trees (BTs)	9.68% (3)	32.26% (10)	41.94% (13)	16.13% (5)	0.00% (0)	31
Finite State Machines (FSMs)	27.27% (3)	36.36% (4)	27.27% (3)	0.00% (0)	9.09% (1)	11
Domain Specific Languages (DSLs)	0.00% (0)	16.67% (1)	0.00% (0)	83.33% (5)	0.00% (0)	6
Planning Domain Definition Language (PDDL)	0.00% (0)	100.00% (1)	0.00% (0)	0.00% (0)	0.00% (0)	1
Hierarchical Task Networks (HTNs)	0.00% (0)	0.00% (0)	0.00% (0)	100.00% (1)	0.00% (0)	1
Markov Decision Processes (or other flavours of Markovian processes)	100.00% (1)	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	1
Hand-coded piece of software with if/else structures in programming language of choice	0.00% (0)	50.00% (1)	0.00% (0)	50.00% (1)	0.00% (0)	2

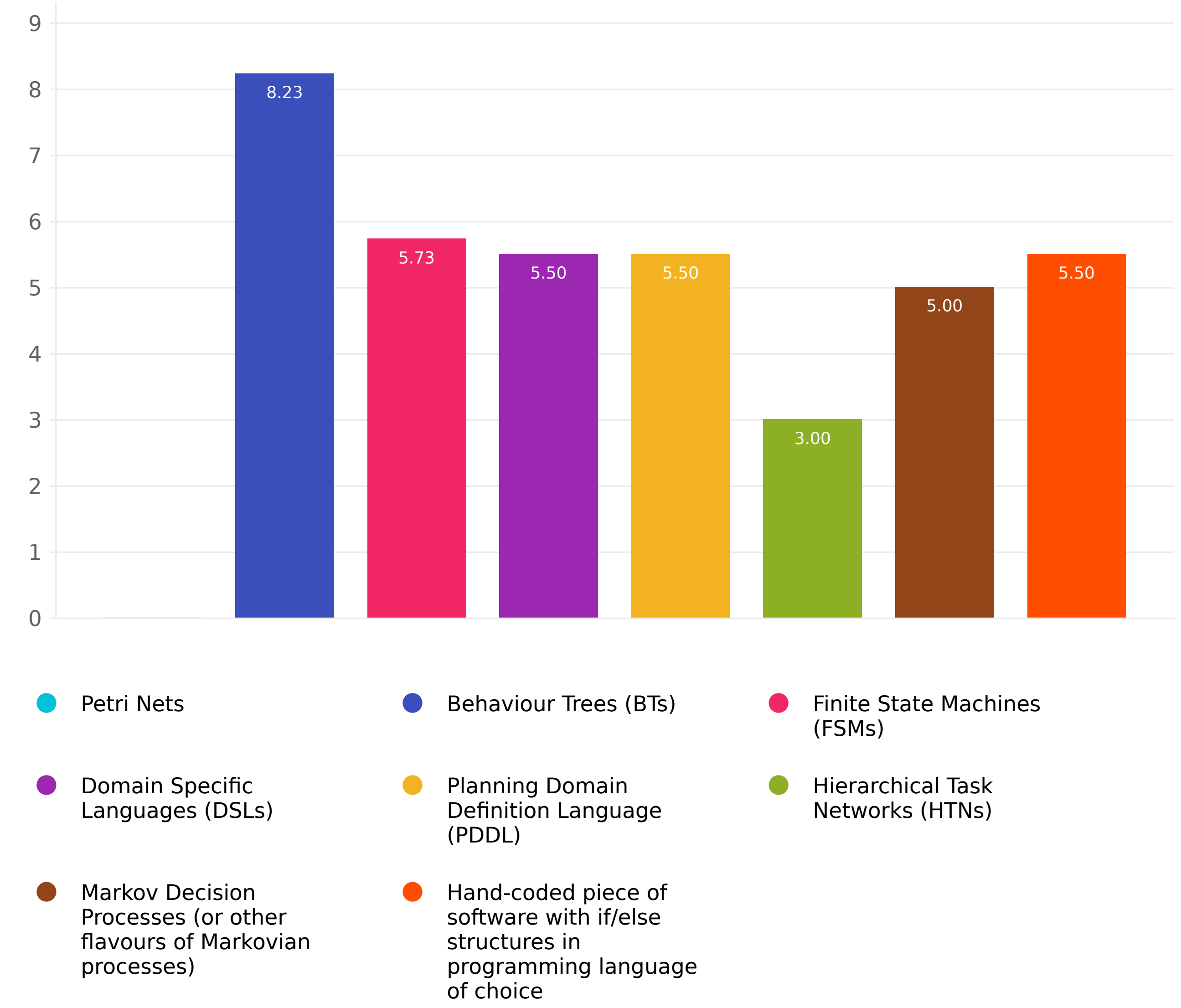
CT3

Which of the following is your *preferred* **deliberation language**?

How keen are you on using Behavior Trees when developing an autonomous system?

1-10 rating scale
0 = Not very keen to use Behavior Trees
10 = Very keen to use Behavior Trees

Answered: 54



Average rating: 7.06

Cross-tab analysis - Systematic testing

CT4

Do you feel the need for a more systematic testing/verification approach within your **deliberation engine**?

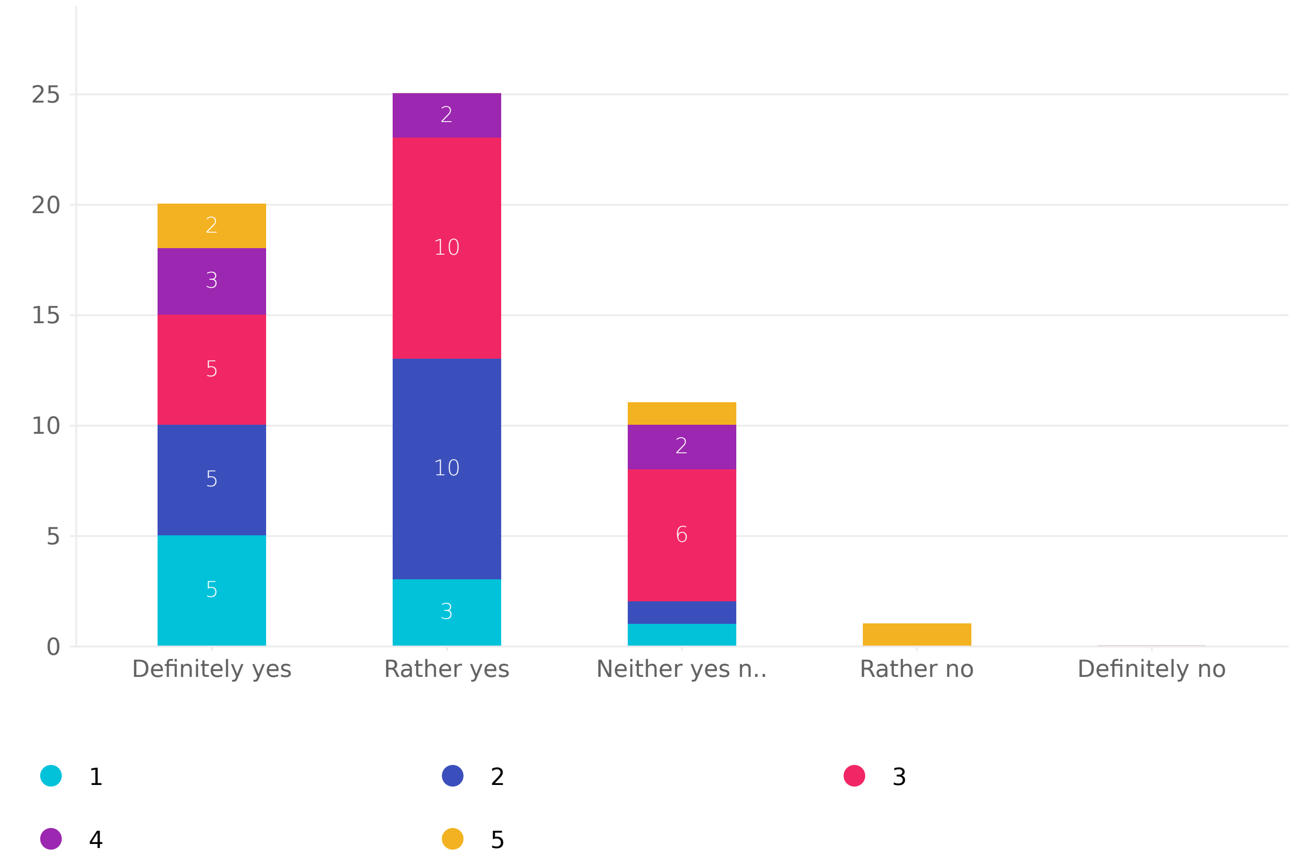
vs

Would you in principle spend effort on writing a formal model of your **deliberation logic** if that provides more systematic testing/verification?

1-5 rating scale
1 = Very willing to spend efforts
5 = Not willing to spend efforts

79% of respondent feel that there is a need for a more systematic testing/verification approach within the deliberation engine, and approximately 50% are willing to spend effort on writing a formal model of the deliberation logic even if that provides more systematic testing/verfication.

Answered: 57



Row	1	2	3	4	5	Average rating	Response count
Definitely yes	25.00% (5)	25.00% (5)	25.00% (5)	15.00% (3)	10.00% (2)	2.6	20
Rather yes	12.00% (3)	40.00% (10)	40.00% (10)	8.00% (2)	0.00% (0)	2.44	25
Neither yes nor no	9.09% (1)	9.09% (1)	54.55% (6)	18.18% (2)	9.09% (1)	3.09	11
Rather no	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	100.00% (1)	5	1
Definitely no	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0.00% (0)	0	0

Average rating: 2.67