LUNDS UNIVERSITET

CHALMERS

EzSkiROS: A Case Study on Embedded Robotics DSLs to Catch Bugs Early

Momina Rizwan* , Ricardo Caldas , Christoph Reichenbach* and Matthias Mayr*

15, May 2023

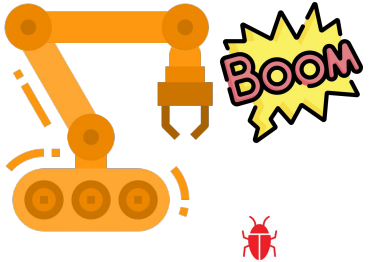
5th International Workshop on
Robotics Software Engineering (RoSE'23)

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

Why finding bugs early is important in robotics?

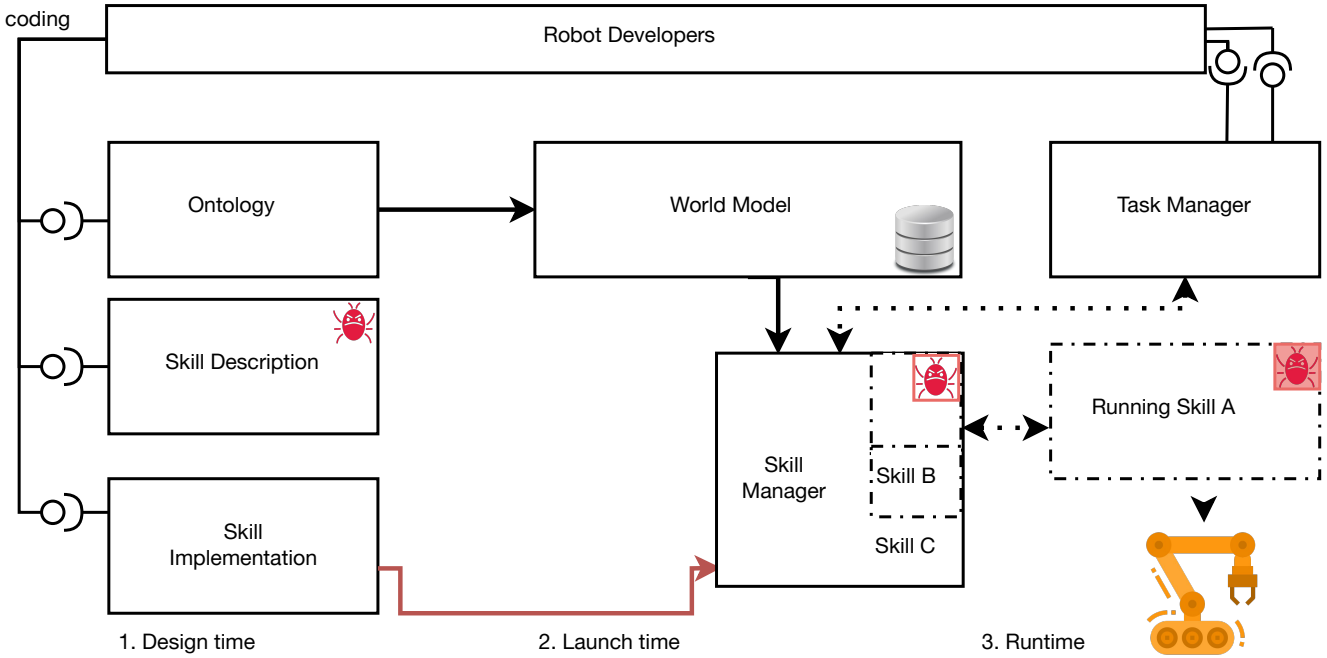
- Hard to predict bugs at development time
- Code usually written in Python
- Bugs are only detected when the bug prone code gets executed.
- Finding errors at runtime can have serious consequences

Design time



Runtime

SkiROS2- A skill based robot execution platform



A Skill Description in SkiROS2

```
class Pick(SkillDescription):
    """
    @brief      Picks an object from a coffee table
    """
    def createDescription(self):
        #=======Params=========
        self.addParam("Robot", Element("rob:Robot"), ParamTypes.Inferred)
        self.addParam("Gripper", Element("rparts:Device"), ParamTypes.Required)
        self.addParam("CoffeeTable", Element("world:Furniture"), ParamTypes.Inferred)


        self.addPreCondition(self.getRelationCond("RobotAtCoffeeTable", "skiros:robotPartOf", "Robot",
"CoffeeTable", True))
```

A Skill Description in SkiROS2

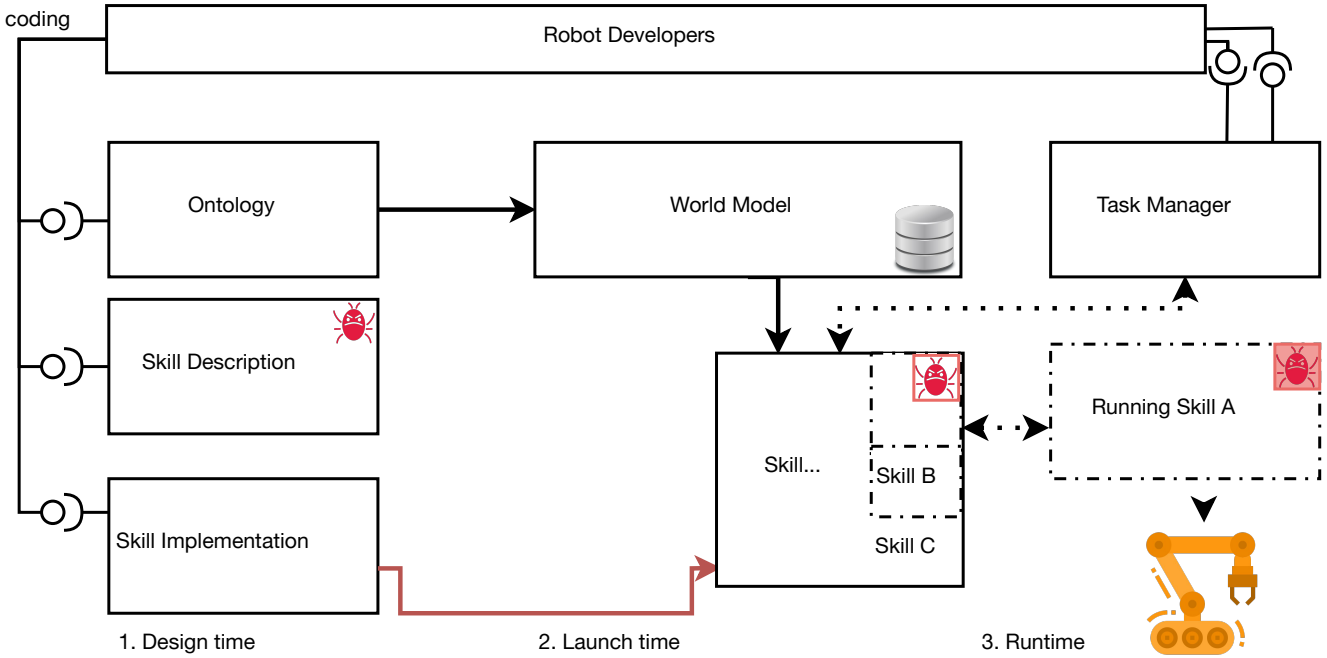
```
class Pick(SkillDescription):
    """
    @brief      Picks an object from a coffee tabl
    """
    def createDescription(self):
        #=======Params=========
        self.addParam("Robot", Element("rob:Robot"), ParamTypes.Inferred)
        self.addParam("Gripper", Element("rparts:Device"), ParamTypes.Required)
        self.addParam("CoffeeTable", Element("world:Furniture"), ParamTypes.Inferred)
```

```
<owl:ObjectProperty rdf:about="http://rvmi.aau.dk/
    ontologies/skiros.#robotPartOf">
    <rdfs:range rdf:resource="#Robot"/>
    <rdfs:domain rdf:resource="#Device"/>
</owl:ObjectProperty>
```

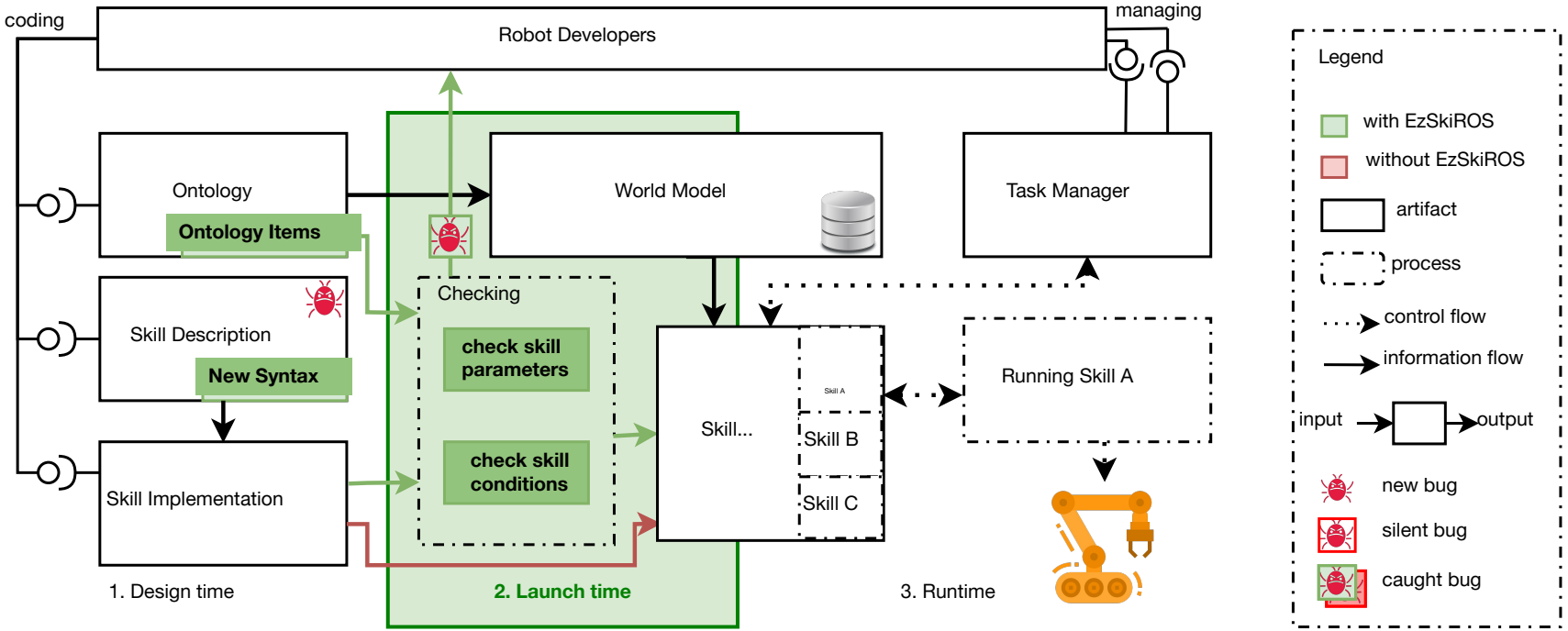
```
        self.addPreCondition(self.getRelationCond("RobotAtCoffeeTable", "skiros:robotPartOf", "Robot",
"CoffeeTable", True))
```

The ontology is not encoded in Python and is linked at launch time.

SkiROS2- A skill based robot execution platform



In this work, we want to report bugs early at launch time.



We propose to embed DSLs directly in Python

- External DSLs require building a frontend
- Internal/Embedded DSL instead reuse an existing “host” language
- Adjusting the language’s behaviour to accommodate the needs of the problem domain.

“The idea behind using an embedded DSL is to implement the error checking code as part of the control logic code.”

Type annotations:

```
def f(x : typeannotation):
```

Overloadable infix operators:

```
class Container(set):
    def __iadd__(self, condition):
        self.add(condition)
        return self


c = Container()
c += 1  # Not allowed on sets in Python
c += 2  # Domain-specific modified syntax
```

Robotics DSL design patterns

- **Domain Language Mapping** – generating classes/types from ontology concepts

```
r = rob.Robot("MyRobotName")
```

Robotics DSL design patterns

- **Domain Language Mapping** – generating classes/types from ontology concepts

- **Early Dynamic Checking** – generating mock code

```
class MyRobotOp(RobotOp):
  ...
  def check(self):  # Checking code
    assert self.config.mode in ["A", "B"]

  def run(self):    # Control logic
    if self.config.mode == "A":
      self.runA();
    elif self.config.mode == "B":
      self.runB(self.config.v + 10);
    else:
      fail()
```

```
class MyRobotOpMock(RobotOp):
  ...
  def check(self):  # Checking code
    assert self.config.mode in ["A", "B"]
    assert isinstance(self.config.v, int)
  def run(self):    # Control logic
    if self.config.mode == "A":
      pass;
    elif self.config.mode == "B":
      pass;
    else:
      fail()
```

Robotics DSL design patterns

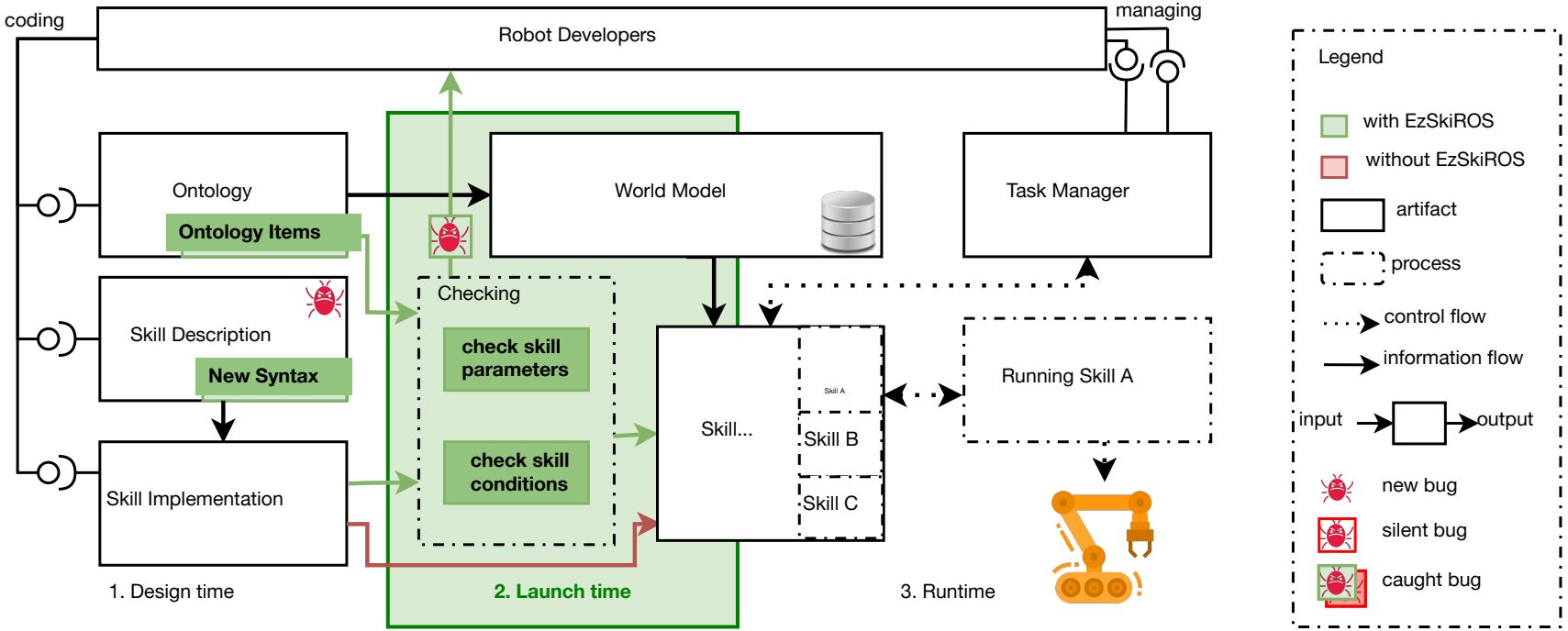
- **Domain Language Mapping** – generating classes/types from ontology concepts
- **Early Dynamic Checking** – generating mock code
- **Symbolic Tracing** - generating mock code for methods with input parameters

```
class SetArmSpeedOp(RobotOp):
  def run(self, speedup):
    self.setArmSpeed(speedup)
    self.setArmSafety(speedup)
```

```
TYPE_CONSTRAINTS = []

class SetArmSpeedOpMock:
  def setArmSpeed(self, speedup):
    TYPE_CONSTRAINTS.append((speedup, float))
  def setArmSafety(self, obj):
    TYPE_CONSTRAINTS.append((speedup, bool))
```

In this work, we report bugs early at launch time ...



Skill Description with EzSkiROS

```
class PickCoffeeTableOp(RobotOp):
  def run(self, robot : rob.Robot, gripper : rparts.Device,
          coffee_table : world.Furniture):

      self.pre_conditions += coffee_table.robotPartOf(robot)
```

- Type annotations are not checked by Python by design.
- We add the functionality to check such types for any mismatch while integrating it with other components.

As a by-product we get a more concise syntax ...

```
class Pick(SkillDescription):
    """
    @brief      Picks an object from a coffee table
    """
    def createDescription(self):
        #=======Params=========
        self.addParam("Robot", Element("rob:Robot"), ParamTypes.Inferred)
        self.addParam("Gripper", Element("rparts:Device"),
ParamTypes.Required)
        self.addParam("CoffeeTable", Element("world:Furniture"),
ParamTypes.Inferred)
        self.addPreCondition(self.getRelationCond("RobotAtCoffeeTable",
"skiros:robotPart", "Robot", "CoffeeTable", True))
```

SkiROS2 syntax

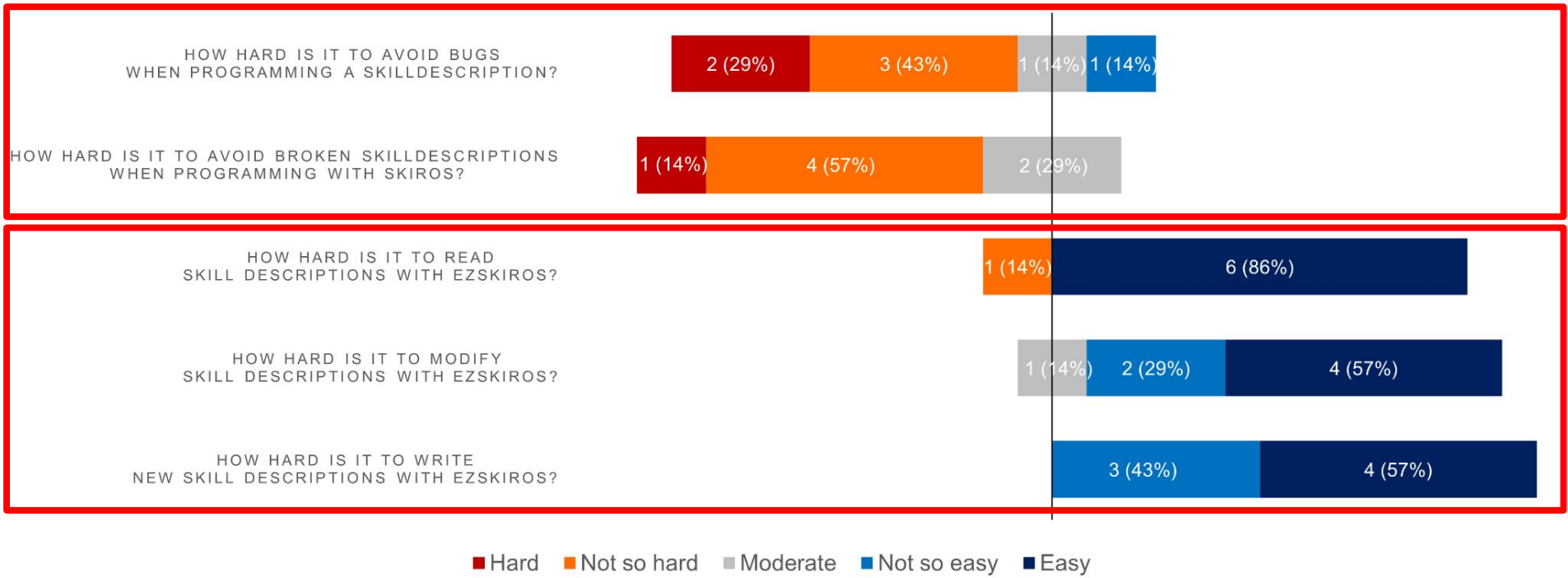
```
class PickCoffeeTableOp(RobotOp):
  def run(self, robot : rob.Robot,
          gripper :rparts.Device,
            coffee_table : world.Furniture):

       self.pre_conditions +=
                          coffee_table.robotPartOf(robot)
```

EzSkiROS syntax

- To evaluate the **effectiveness** of our approach, we did an initial user study with **7** SkiROS developers from academia and industry.
- We showed a video demonstration of the tool.
- After that, participants answered a survey.

User study Results with SkiROS2 Developers



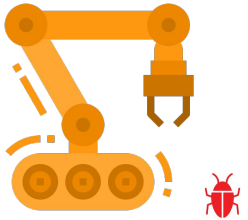
*The questions in Y-axis were altered for illustration purposes. Check our paper for the original questions.

Conclusion

- We describe DSL design patterns suitable for the robotics domain and demonstrate our approach for DSL embedding in Python,
- We implement EzSkiROS, a DSL for early detection of type and configuration errors.

Design time

Launch time



Runtime

More details in the paper about

"How you can adapt these patterns to your own purposes."

For further questions, you can contact me at momina.rizwan@cs.lth.se !