


An Analysis of Behaviour-Driven Requirement Specification for Robotic Competitions

Minh Nguyen^{*†} 

Nico Hochgeschwender^{*} 

Sebastian Wrede[†] 

Abstract—Recent scientific and technological advances have enabled robotic applications in various challenging domains, which motivates means to better represent and manage the subsequent increase in number and complexity of requirements. We look into rulebooks of robotic competitions and benchmarks as one publicly available source of requirements and acceptance criteria for evaluating robots’ performance. From our analysis, we derive a Feature Model containing common elements that recur in descriptions of different robotic competitions. We argue how these *features* can be used to express requirements and acceptance criteria for robotic applications, within the context of the Behaviour-Driven Development (BDD) paradigm. This can serve as a mean not only to analyse and manage requirements, but also to introduce automation into verifying and validating requirements in robotics.

Index Terms—Robotics, Requirements, Model-Driven Engineering, Behaviour-Driven Development

I. INTRODUCTION

The development of any complex system typically involves a close interplay between requirements definition and system design. Starting with a set of abstract requirements, engineers identify design constraints and make decisions to shape the initial system architecture. This necessitates the refinement of the original requirements to be more specific, which in turn leads to constraints and decisions that further shape the system design as the development process progresses.

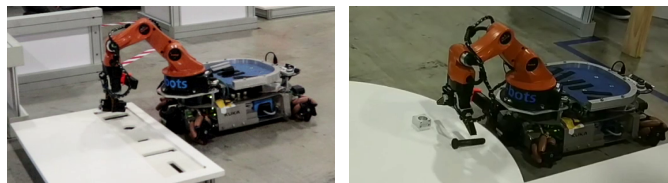
In recent years, robotic systems have emerged beyond factory floors into domains such as healthcare and agriculture, where they must face complex and unpredictable environments while performing increasingly elaborate behaviours, sometimes involving interactions with humans and other robots. Representing and managing requirements for such applications can be exceedingly difficult, because of both their sheer number and the underlying knowledge from interleaving, complex domains. In order to tackle this challenge, we investigate the following research questions in this paper:

- Which methodologies and concepts are used for requirements specification in robotics competitions?
- How are the concepts of behaviour-driven development applicable to specify robotics applications?
- What are the challenges for automatic verification of acceptance criteria in a robotics context?

^{*} Hochschule Bonn-Rhein-Sieg, Germany. {minh.nguyen, nico.hochgeschwender}@h-brs.de

[†] Bielefeld University, Germany. sebastian.wrede@uni-bielefeld.de

This work is supported in part by the European Union’s Horizon 2020 project SESAME (Grant nr. No 101017258).



(a) Precision Placement Test (PPT)

(b) Rotating Table Test (RTT)

Fig. 1: Images from official runs at the 2019 RoboCup@Work World Cup in Sydney.

Towards finding answers for these questions, we first investigate in Section II the current practice of requirement specification in robotics. As requirement-related artefacts for commercial robotics applications are not easily accessible, we focus our effort on publicly available sources for robotic requirements, namely scientific publications, technical standards and rulebooks for robotic competitions and benchmarks. Among these, competition rulebooks are of particular interest because they describe not only the scenarios for verifying the robotic systems against the specified requirements, but also the criteria for evaluating the robots’ performance in carrying out these scenarios. Drawing on direct experience with designing, organizing and competing in many of these events [1]–[4], we analyse these rulebooks to identify common concepts used to express robotic requirements and present them in Section III in the form of a Feature Model [5], as shown in Fig. 2. In Section IV, we then investigate how these concepts can be used to represent robotic requirements in an executable format, specifically with Behaviour-Driven Development (BDD) [6], a popular approach for acceptance testing. Finally, we discuss in Section V the future steps towards automating the execution of these BDD specifications.

II. BACKGROUND

A. Related Work

Bozhinoski et al. [11] survey robotic literature concerning safety management for mobile robotic systems and present a classification scheme for scientific publications on the topic. While the study provides interesting insights and valuable references for safety-related robotic requirements, its focus is not in how such requirements are specified.

Other studies also analysed competitions and benchmarks to gain insights about robotic applications. Sun et al. [8] investigate grasping and manipulation tasks included in these

TABLE I: Robotics competitions and benchmarks considered in this study.

Competition	Abbreviation	Event Years	Rulebook Year
RoboCup@Home [4]	HOME	2006–2022	2022
RoboCup@Work [2]	WORK	2014–2022	2022
Evaluation methodology for healthcare competitions (HEART-MET) [3]	HEART	2020–2022	2020
Robotic Grasping and Manipulation Competition [7][8]	RGMC	2016–2022	2018
Robot Competitions Kick Innovation (RoCKIn) [1]	RoCKIn	2014, 2015	2015
Soft Manipulation Project (SOMA) Bin-Picking Benchmark [9]	SOMA	2019*	2019
Australian Centre for Robotic Vision (ACRV) Picking Benchmark [10]	ACRV	2017*	2017

* No public competition organized.

events to identify key challenges and future research directions for the robotic manipulation domain, which then guide iterations of the task pool in Robot Grasping and Manipulation Competitions (RGMC). Brancalião et al. [12] surveys mobile robot competitions to investigate their educational impact via analysing the events’ objectives and addressed challenges, their final applications areas, their target audience, as well as the technologies utilized at the competitions.

A recent study [13] examine experience reports from and interviews with robotics companies to identify three “drivers” of variability in the service robotics domain and investigate how these companies manage them. While the variability drivers, namely environment, robot hardware, and mission, resemble respective *features* discussed in Section III, the work in [13] has a different focus from our work and does not explicitly consider requirements and their verification.

B. Sources of Requirements in Robotics

To the best of our knowledge, there are no repositories of requirements publicly available in robotics. Studies looking into private sources of requirements [13][14] typically rely on experience reports and employees interviews that are difficult to be done on a broader scale. Instead, we identify three other open sources of robotic requirements, namely *standards*, *research in scenario modelling* and *scientific competitions*.

1) *Standards*: Depending on the particular application domain for which the robotic system is developed, robots must often adhere to one or more standards. Delivery robots that traverse side walks and bike baths or autonomous cars, for example, must comply to country-specific traffic regulations. Few standards are available which directly address robotic applications [15]. Most notably, ISO/TS 15066:2016¹ is a safety standard on industrial robots which extends ISO 10218-1² and ISO 10218-2³ for applications involving collaborative robots. As standards must apply to a wide range of systems, specification of requirements in these documents are typically at a high level and do not specify how verification of these requirements can be carried out. Instead, it is up to test engineers to define

test scenarios that, i.e. for collaborative robots, consider the relevant methods for safety assessment [16] and associated acceptance criteria to verify application requirements.

2) *Research in Scenario Modelling*: Modelling a scenario to evaluate robot performance is an important step in developing robotic applications, which typically involves the specification of some functional aspects of the associated requirements [13][17]. For example, Knoop et al. [18] identify “abstract task knowledge” for mapping to robot execution system, which includes abstractions of robot behaviours and task-specific constraints on the environment. As pointed out in [17], few approaches exist that tackle the challenge of modelling robot scenarios, and of the ones included in the paper, none explicitly addresses verification of robotic requirements.

3) *Robotics Competition Rulebooks*: Yet another public source of requirement specifications in robotics are rulebooks of competitions and benchmarks. Compared to the sources described above, rulebooks specify test scenarios at a concrete level, while also include descriptions of how to evaluate robots’ performance of such scenarios. Table I lists the competitions and benchmarks considered in this paper. We focus our study on ground-based competitions with manipulation tasks and prioritize ones involving mobile robots and interactions with human actors. As such, we omit competitions that do not align with our focus, e.g. the Logistics League⁴ at RoboCup. We also omit SciRoc⁵ and other European Robotics League (ERL)⁶ competitions because of significant overlap with events already included in Table I.

III. ANALYSIS OF REQUIREMENT SPECIFICATION IN SCIENTIFIC ROBOTIC COMPETITIONS

In general, rulebooks for robotic benchmarks and competitions must define the *Scenarios* in which the robots must partake and *Evaluation Criteria* for assessing their performance. Optionally, the design of the scenario and evaluation plan may depend on an overall *Objective* of the competition, i.e. a particular scientific or technological challenge that the competition aims to address. In this section, we review the rulebooks

¹<https://www.iso.org/standard/62996.html>

²<https://www.iso.org/standard/51330.html>

³<https://www.iso.org/standard/41571.html>

⁴<https://ll.robocup.org/>

⁵<https://sciroc.org/>

⁶<https://eu-robotics.net/eurobotics/activities/european-robotics-league/>

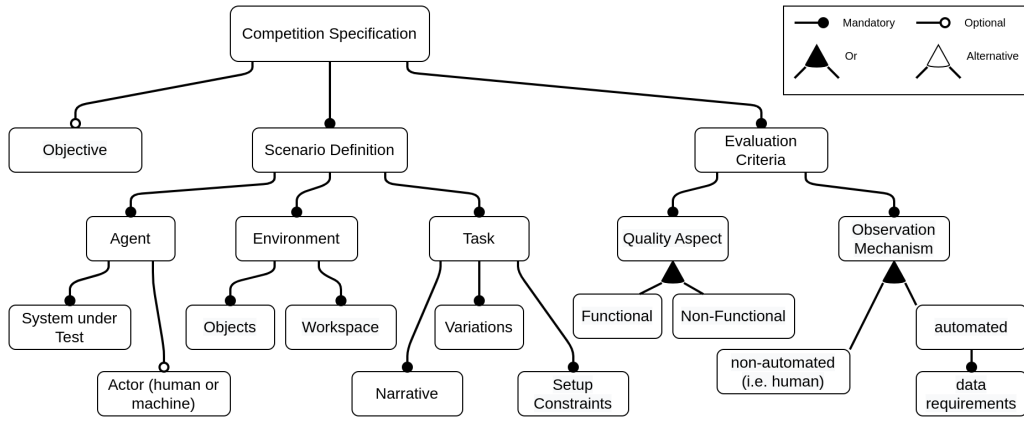


Fig. 2: A feature model for rulebooks of robotics competitions.

of the events listed in Table I to further identify common concepts, i.e. “features,” used to specify these benchmarks and competitions and visualize our analysis in a Feature Model [5], as shown in Fig. 2.

A. Competition Objective

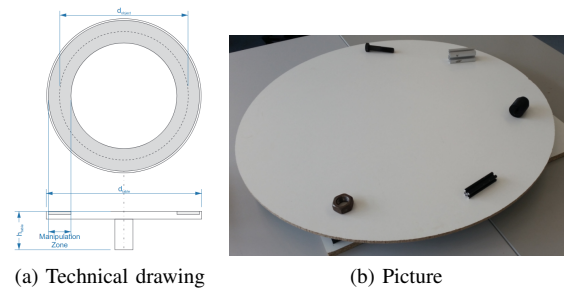
As competitions and benchmarks typically approximate a real robotic application to evaluate robot performance, they often require a general **Objective** to guide these approximations, influencing how scenarios are defined and evaluated. Such objectives may address specific research or technological challenges, or target specific application domains. For example, the RoCKIn competitions, SOMA and ACRV benchmarks aim to promote *reproducibility* of robot performance evaluation, whereas HEART-MET targets “tasks relevant to healthcare settings.” Some competitions also state more general, long-term objectives that hold less impact over the competition design, such as the RoboCup’s ultimate goal for a team of autonomous robots to “win a soccer game [...] against the winner of the most recent World Cup⁷.”

B. Scenario Definition

To compare and evaluate robot performance, rulebooks must describe how the **Scenario** should unfold. Such descriptions typically include information about the **Environment** in which the scenario takes place, the **Agents** involved, and the **Task** that the system under evaluation should perform.

1) *Environment*: Rulebooks’ specifications of the environment, in general, either describe the physical **Objects** that the robot(s) may interact with, or abstract **Workspaces** in which they may operate. Fig. 3 shows several formats employed to specify a rotating table in RoboCup@Work [2]. More environment specification examples can be found in Table II.

Objects are physical bodies in the environment with which the robotic system(s) is expected to interact while performing some task. Here, interactions can vary from simply transporting an object from one place to another, to complex, contextual manipulation tasks like opening a fridge door to



same table speed. Example: For a Rotating Table with diameter $d_{table} = 1\text{m}$, Objects are placed on a grasp region with the diameter $d_{object} = 0.8\text{m}$ with $\omega_{table} = \frac{2 \cdot d_{object}}{d_{grasp}} = \frac{2 \cdot 0.2}{0.8} = 0.5\text{rad s}^{-1}$ and with $n_{table} = \frac{\omega_{table}}{2\pi} = \frac{0.5}{2\pi}$ the minimum rotational speed of the Rotating Table $n_{table} = 0.0796\text{s}^{-1}$

(c) Text and equations.

Fig. 3: Different specification formats for a rotating table [2].

look for food items [4]. In addition to the task, the choice of objects may also depend on the objective the competition tries to address. In [9] for example, to promote “comparability,” five object categories of varying difficulty levels are proposed to target known manipulation and perception challenges like soft bodies or transparent objects. Robots’ performance can then be associated with the object category chosen for their specific execution, which allows for a better comparison across different robotic systems and events.

Workspaces are abstract segmentations of the robot’s operational space that hold distinct semantics in a task. For competitions involving stationary robot arms [7][9][10], the workspace is limited by the arm’s reachability around its mounting location. Description of workspaces, in this case, can specify constraints on objects to ensure that they are reachable for the platforms expected at the competition. With mobile robots, on the other hand, rulebooks may also describe operating areas between which the robot(s) may have to navigate. Examples include rooms in a flat [3][4] or “service areas” equipped for specific tasks [2]. Here, rulebooks may specify dimensions and functions of objects expected to be in the workspaces, e.g. tools in “spatial areas” in [1], or constraints on the pathway connecting these areas to ensure

⁷<https://www.robocup.org/objective>

TABLE II: Exemplary environment descriptions in rulebooks

Comp.	Objects	Workspace
HOME	Custom domestic objects	Rooms in a typical flat (kitchen, living room, ...)
WORK	Custom, industrial-like objects (screws, nuts, ...)	“Service areas” for “loading,” with “rotating tables”
HEART	“General domestic & healthcare-related objects”	Rooms in a typical flat
RGMC	YCB ⁸ , APC2015 ⁹	Consists “table,” “basket,” ...
RoCKIn	grouped by expected robot behaviour.	“Spatial areas”
SOMA	“Fruits and vegetables products” (including packaging)	“Positioning of robot and containers depends on reachability and workspace of the robotic arm”
ACRV	YCB, custom objects	“tote is positioned within a 2 m workspace in front of the shelf”

that robots can move freely between them [2]. Competition environments can also comprise a hierarchy of workspaces. In RoboCup@Home [4], for instance, the robot may be tasked with bringing dishes from the table in the dining room to the dishwasher in the kitchen. Here, “table” and the “dishwasher” characterize workspaces that are contained in the dining room and the kitchen, respectively.

2) *Agents*: Distinct from passive elements of the environment, competitions and benchmarks also include *Agents* which can independently and actively exhibit behaviours in a task. Most obvious are the *System(s) under Test (SUT)*, i.e. the robots being evaluated in the competition. Description of SUT in the rulebooks may include expected capabilities and safety features, e.g. having an emergency stop button, or constraints on physical dimensions to ensure that they can move freely in the competition arena. Additionally, some competitions require the SUT to interact with other agents, here termed *Actors*. These interactions typically involve the actors providing information or objects necessary for the SUT to carry out the task. In tasks involving interactions with humans, e.g. in [3] and [4], the missing information can be behaviours that the robots must detect visually, or verbal commands that must be acquired via speech recognition. The “Hand Me That” task in [4], for example, requires the robot to detect and obtain an object that the human actor is gesturing for. Apart from humans, actors can also be other autonomous systems, e.g. in [2] and [3], where the robots get task specifications from and send feedback messages to an automated “referee box”.

3) *Task*: Descriptions of *Tasks* in a competition typically include information about the *Narrative* of how the scenario should play out, the *Setup Constraints* necessary for task execution and evaluation, and the task *Variations* for properly evaluating the robot performance.

A scenario *Narrative* describes how the “scenes” of the task

⁸Yale-CMU-Berkeley (YCB) object and model set. Online: <https://www.ycbbenchmarks.com/object-set/>.

⁹2015 Amazon Picking Challenge object set. Online: https://rll.berkeley.edu/amazon_picking_challenge/

should unfold, typically as an outline of the robots’ expected behaviours and their interactions with actors during task execution. The “Object Handover” task in [3], for example, includes step-by-step descriptions of how the robot should communicate with the “referee box” and interact with the human actor. Aside from procedural descriptions, rulebooks may also specify objectives to be fulfilled by the end of the task. In the “Storing Groceries” task in RoboCup@Home, for instance, an objective is to group the objects by some categories, on top of the narrative to transport objects from the table to the cabinet.

Depending on the evaluation plan and the specific challenge being addressed, competitions and benchmarks often introduce *Variations* of the task for different executions. For example, to promote reproducibility and comparability in evaluating performance of the pick-and-place task, both the ACRV [10] and SOMA [9] benchmarks introduce variations of the task difficulty in slightly different manners. While both utilize object cluttering as a factor for assigning difficulty levels, [10] focuses on the reproducibility of geometric configurations of the objects, whereas [9], driven by research interest in soft manipulation, characterizes their task difficulty by classes of challenging objects, e.g. articulated or deformable bodies. In addition to environment aspects, rulebooks may also specify variations of the actors’ behaviours. The “Object Handover” task in HEART-MET [3], for instance, evaluates the robots’ ability to adapt its behaviour based on whether the human actor is standing, sitting or lying down.

Finally, task narratives are typically accompanied by *Setup Constraints* on various aspects of the environment and agents to ensure both feasibility of the task and proper evaluation of the robots’ performance. Such constraints may simply list which objects or agents are required to be in the workspace, e.g. a container for pouring [3][4][7] or a person to hand an object to [3][4]. Constraints may also address task feasibility by accommodating expected hardware limitations of the robots, e.g. materials visible to perceptual sensors [2]. In addition to feasibility, constraints may also ensure fair evaluation of robots’ performance by specifying some properties that should remain invariant across different executions. The “Where is This?” task in [4], for instance, requires the human “operator” to be “non-expert,” i.e. an audience member with no robotics background, to make sure that their communication with the robots is not biased by “expert” knowledge.

C. Evaluation Criteria

Having defined the scenarios, rulebooks then describe how to evaluate robots’ performance as they carry out the specified tasks. *Evaluation Criteria* in rulebooks are typically characterized by the *Quality Aspect* that they aim to evaluate, and the *Observation Mechanism* through which the evaluation result can be obtained.

1) *Quality Aspect*: Evaluation criteria aim to assess some *Quality Aspect* of the systems’ performance, which can be *functional* or *non-functional*. We base this distinction on the terminology in requirements engineering, where functional

TABLE III: Common evaluation criteria in robot benchmarks and competitions

Quality Aspect	Examples	Competition	Observation Mechanism ^a
Functional			
Success criteria	Item is in the work order, not dropped from more than 35 cm when placed in the tote “Object is transported out of the corresponding Service Area”	ACRV WORK	Mn Mn
Failure criteria, i.e. terminating conditions	“Robot damages or destroys the manipulation objects” “May not take more than 15 minutes to fulfil the full work order”	RoCKIn ACRV	Mn Mn
Partial success or failure	<i>Hammer a Nail</i> task: 2 points for driving the nail up to 5.1 cm (5 points for full depth) −100 points for “object being dropped to the floor” “reduction of 10% of the maximum score when the robot request a partial solution”	RGMC WORK HOME	Mn Mn Mn
Non-functional			
Robustness	Variations of <i>success rate</i> , i.e. successful executions over total number of attempts	SOMA, ACRV	Mn
Execution time	Bonus points for finishing scenario with extra time Average and standard deviation of duration of a successful pick-and-place cycle	WORK SOMA	Am Mn
Compositions			
i.e. combinations of multiple criteria	<i>Storing Groceries</i> task: 50 points for each successful transportation of 5 objects, 90 points for each successful categorization.	HOME	Mn

^a Observation Mechanism: Am: Automated, Mn: Manual.

requirements are ones that concern the expected behaviour of the system, i.e. describing *what* the system shall do, whereas non-functional requirements address any other quality aspects that are not covered by functional requirements [19].

Here, we consider an evaluation criterion to be functional if it assesses whether the SUT is exhibiting its expected behaviour, i.e. establishing what constitute the SUT’s behaviour to be “correct” or “incorrect” compared to what is expected. An example of a “correctness” criterion can be found in manipulation tasks of the RoboCup@Work competition, where an object is considered successfully picked up if the robot “transports it out of the corresponding Service Area,” which has “infinite height.” Criteria for identifying “incorrect” behaviour may determine that the SUT’s behaviour has diverged so far from expectation for the task to be feasible, is violating some critical safety constraints, or that the execution time has exceeded a predefined limit. Examples include knocking over objects [7] or uncontrolled collision with the environment or human actors [3][4]. Additionally, “partially correct” criteria allow for tolerance of some “incorrectness,” usually in the form of partial scores or penalties. The RoboCup@Home competition, for instance, allows the assistance of human actors for some of its tasks in exchange for some reduction of the total scores given to the robot’s performance.

While non-functional criteria do not directly address the “correctness” of system behaviours, they rely on knowing whether the behaviour is correct to evaluate other quality aspects. For example, several competitions [10][9] employ “success rate” to evaluate robot performance, typically as the ratio between the number of successful executions over the total number of attempts. This criterion characterizes the *robustness* of the system behaviour, but it relies on the criteria that classify whether an execution is “successful” or not.

Finally, robot performance may be evaluated using a combination of functional and non-functional criteria. This is especially common at robot competitions [2][3][4], where tasks may involve compositions of robot behaviours. Here,

scores are typically accumulated from several functional and non-functional criteria to calculate an overall performance score at the end of each execution.

2) *Observation Mechanism*: Rulebooks also specify mechanisms through which evaluation criteria can be applied to draw conclusions about the robots’ performance. Most competitions and benchmarks employ human judges to carry out the evaluation process. Even with detailed descriptions of the criteria, the evaluation in this case still depends on the subjective view of the judges. RoboCup@Work, for example, has specific criteria for what constitutes a valid picking behaviour, but scoring is done by human referees observing the robots from different perspectives. HEART-MET, RoboCup@Work, and RoCKIn utilize an automated “referee box” in the evaluation process, but only to measure execution time to give bonus points for finishing the task early.

IV. BEHAVIOUR-DRIVEN DEVELOPMENT FOR ROBOTICS

Behaviour-Driven Development (BDD) [6] is a popular acceptance testing approach, which extends Test Driven Development (TDD) [20] by formulating the acceptance criteria for each **Feature** as a list of **Scenario**’s that capture the expected behaviours of the System under Test (SUT). Each **Scenario** is specified using the semiformal construct: **Given** [initial conditions], **When** [an event occurs], **Then** [ensure outcomes]. Most notable among BDD implementations is the Cucumber¹⁰ toolchain, which introduces the Gherkin syntax with support for automatic execution available in Java, C++, Ruby, Python and many other languages. Gherkin extends the original BDD formulation with additional keywords like **Background** and **Examples** to ease the transformation of acceptance criteria into executable BDD test cases. While such tooling and automation support for BDD has been successfully leveraged in many other domains, its adoption in robotics remains limited to a few research articles. Of the identified publications, [21] and [22] employ BDD to

¹⁰<https://cucumber.io/>

enable formal verification of some execution model of Cyber-Physical Systems (CPS), whereas Deng et al. [23] uses BDD scenarios to describe metamorphic relations of driving models in autonomous driving.

In this section, we investigate how we can leverage the BDD methodology and tooling for specifying robotic requirements. To ground the discussion in a concrete example, we consider a pick-and-place behaviour, which appears in most competitions and benchmarks. We first discuss how BDD scenarios for one variant of this behaviour, e.g. the “Basic Manipulation Test” at the RoboCup@Work competition [2], can be formulated using concepts identified in Section III. Here, we also pinpoint aspects of the acceptance criteria that remain difficult to represent using the BDD methodology, even with the extensions introduced in Gherkin. Afterwards, we identify challenges with formulating BDD scenarios for different variants of the same pick-and-place behaviour from various rulebooks.

A. Applying BDD to One Pick-and-Place Variant

Listing 1 shows how a BDD **Feature** for the pick-and-place behaviour in the RoboCup@Work competition may be formulated using the Gherkin syntax. We observe that the rulebook’s instantiations of concepts described in Section III can be used to populate the BDD clauses in our example.

First, BDD features typically begin with a user story in the form **As** [role] **I Want** [feature] **So That** [benefit]. In Listing 1, the user story is specified using the *Narrative* of the Basic Manipulation Task (BMT) in the RoboCup@Work rulebook. Next in this example is the **Background** section containing **Given** clauses, which is an extension of the original BDD formulation [6] introduced by Gherkin to specify conditions common to multiple **Scenarios** of a **Feature**. Here, the section is used to specify attributes of the *Environment* that can be referred to later on in the scenarios, e.g. when a **Then** clause evaluates whether an object is heavy in order to give bonus points.

In Listing 1, the pick-and-place behaviour is decomposed into two distinct scenarios, respectively, for the picking and placing portions of the task. In the **pickup** scenario, the **Given** clause constraining the object’s position at the service area is a *Setup Constraint* based on the reachability of the competition’s standard robot platform. The **When** clauses in both scenarios describe events which trigger the picking and placing behaviours, which in this case refer to capabilities of the SUT to detect objects and localize in the environment. The **Then** clauses in the scenarios specify the *functional Evaluation Criteria* for determining the validity of a picking or placing behaviour, which serves as the basis for scoring the robot’s performance throughout the task. Gherkin also introduced **Examples**, which works in conjunction with **Scenario Outline** to substitute terms surrounded by angle brackets (<>) by values listed in the tables accompanying each scenario. In Listing 1, this allows for *Variations* of the objects and the service areas for the pick-and-place task.

Even with the additional keywords introduced by Gherkin, not all information relevant to describe robotic acceptance

```

1 Feature: basic manipulation test
2
3 As A Competitor
4 I Want the robot to transport some objects
5         between two service areas
6 So That I can test basic manipulation
7         functionalities of the robot
8
9 Background:
10 Given a set of objects
11     | ID      | Mass_g |
12     | F20_20_B | 49     |
13     | Axis     | 40     |
14 And service areas
15     | ID | Type |
16     | P1 | Plane |
17     | P2 | Plane |
18     | S1 | Shelf |
19
20 Scenario Outline: pickup
21 Given "<object>" is between 2cm and 20cm
22     from the edge of "<service area>"
23 When "<object>" is located
24 Then "<object>" is out of "<service area>"
25
26 Examples: Objects and service areas
27     | object | service area |
28     | F20_20_B | P2           |
29     | Axis     | P1           |
30
31 Scenario Outline: place
32 Given "<object>" is picked up by the robot
33 When the "<service area>" is reachable
34 Then "<object>" is touching the surface of
35     "<service area>"
36 And "<object>" is not moving
37     at the end of the run
38 And the robot does not drop "<object>"
39
40 Examples: objects
41     | object | service area |
42     | F20_20_B | S1           |
43     | Axis     | P2           |

```

Listing 1: A potential BDD specification in the Gherkin syntax for the pick-and-place behaviour in the RoboCup@Work competition [2]. The navigation between service areas is omitted for simplicity.

criteria can be easily captured by the BDD formulation. In Listing 1, the pick-and-place behaviour is decomposed into separate scenarios for picking and placing. This implies the successful completion of the first scenario is a precondition for the second scenario, e.g. the clause **Given** "<object>" is picked up..., which is information available only at runtime. In addition to such causal relations between scenarios, a scenario in one **Feature** may be reused in another. For instance, the HEART-MET competition [3] has a “Transport Drink” task which involves the pick-and-place behaviour similar to the one described in Listing 1. The competition also has two other tasks, however, in which the robot has to either hand the object to another person or pour its content into a cup, respectively. Here, the same picking scenario is coupled with different scenarios for each behaviour, which would likely result in repetition in current BDD implementations.

Furthermore, while Gherkin provides the **Background** keyword for shared preconditions within a **Feature**, this does not account for ones which are common for multiple

TABLE IV: Some common robot behaviours in robot competitions. \checkmark^* denotes that multiple variations of the same behaviour are used at the competitions.

Behaviour	HOME	WORK	HEART	SOMA	RGMC	ACRV
Pick	\checkmark	\checkmark^*	\checkmark	\checkmark	\checkmark^*	\checkmark
Place	\checkmark^*	\checkmark^*	\checkmark^*	\checkmark	\checkmark^*	\checkmark
Navigate	\checkmark	\checkmark	\checkmark			
Pour	\checkmark		\checkmark		\checkmark	
Hand-over	\checkmark		\checkmark			

Feature's. Objects, for example, can be relevant across all tasks in a competition, or even across competitions in the case of standard sets like the YCB object set. Additionally, **Background** does not include **Then** clauses for evaluation criteria that may apply to multiple scenarios, such as ones for valid picking and placing behaviour shown in Listing 1.

Finally, one important aspect of “behaviours” not explicitly addressed in the BDD formulation is the duration of their execution. This is distinct from what is specified by the **When** clause, which signifies the beginning of the behaviour. In this case, evaluation criteria of robot performance often imply some timing constraints on when they should be checked. For instance, the criterion `object is out of service area` in Listing 1 should be checked only after the picking behaviour has completed. Such constraints can characterize the boundaries of time durations, e.g. “before,” “within,” “after,” or denote intervals, e.g. “every 30 seconds.” While expressing these constraints as texts and numbers is possible in BDD tools, the temporal semantics they hold would be lost, which can hinder the specification of constraints that apply to multiple scenarios. An example is the overall time limit of the Basic Manipulation Task shown in Listing 1 which would encompass timing requirements for the both picking and placing behaviours.

B. Applying BDD to Different Pick-and-Place Variants

As listed in Table IV, tasks in robotic benchmarks and competitions often involve similar robot behaviours. The pick-and-place behaviour, for example, appears in all competitions in Table I. RoboCup@Work includes multiple variations of this behaviour, where the robot must pick from a rotating table or place in cavities tailored to object shapes (shown in Fig. 1). This motivates the formulation of more abstract BDD scenarios that can accommodate the different variability dimensions of expected robot behaviours, i.e. acceptance criteria, as well as a mechanism to introduce these variations when considering a specific task or competition.

The Feature Model described in Section III can serve as the basis for introducing such abstractions to BDD scenarios. Several *features* in Listing 1, for example, can be replaced with more general versions that can accommodate variations of the task in different rulebooks. Specifically, consider a more abstract *Setup Constraint* for the picking behaviour, `target object is reachable`, and more abstract *Evaluation Cri-*

teria, `target object is picked up` and `target object is placed`, for evaluating the robots’ performance. Table V shows how different rulebooks may describe variations of these abstractions. While abstracting BDD scenarios can allow them to cope with variability in rulebooks’ descriptions, evaluating a particular variation of the task requires the variation-specific information to be reintroduced. Designing and implementing a mechanism to do this remains open for future work.

V. TOWARDS EXECUTABLE ROBOTIC SCENARIOS

In Section IV, we discussed how features identified in Section III can be used to formulate BDD scenarios that capture requirements in robotic competitions and benchmarks. To make these BDD scenarios executable, a traditional approach would require significant manual implementation of test code to verify each **Given**, **When**, **Then** clauses. These implementations require runtime information available only during test execution and may not be easily expressed using BDD.

Consider the `pickup` scenario in Listing 1. Such runtime information includes the geometric configurations of objects relative to each other before and after the picking behaviour for checking the **Given** and **Then** clauses. This requires a mechanism to monitor measurements of relevant geometric quantities throughout execution of the behaviour.

Additional timing information about the behaviour is also necessary to trigger transitions between the scenario clauses and to indicate *when* to evaluate the outcome of the behaviour, i.e. **Then** clauses. In Listing 1, only information about when the behaviour begins, e.g. `object is located`, is available. An executable scenario, therefore, would require both explicit specification of such timing information and communication of this information from the robots to the test process.

Such measurement and communication mechanisms must cope with various variability dimensions introduced by different robotic platforms, each having different software and hardware capabilities. Furthermore, competitions like RoboCup@Home and RoboCup@Work also offer simulation leagues which include the same tasks used in real world events. In this case, the semantics of the runtime information required does not change, but the mechanisms used to acquire such information may differ compared to in the real world.

Notably, via the use of the “referee box,” the HEART-MET [3] competition requires the robot to provide certain runtime information about its performance. For example, in the object handover task, the robot must communicate its estimation of the human actor’s pose and whether the person successfully grasped the object. While this mechanism must depend on the accuracy of the robots’ estimation, it partially enables the automatic evaluation of the robots’ performance and can serve as inspiration for a more general approach.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we analyse rulebooks of robotic competitions and benchmarks as a publicly available source of requirements for robotic applications. From our analysis, we identify concepts recurring in descriptions of these events and present them

TABLE V: Variations of specification in different rulebooks for the same abstract criteria.

Variation	target object is reachable	target object is picked up	target object is placed
WORK BMT ^a	Manipulation zone of service area: <ul style="list-style-type: none"> • minimum depth is 20 cm • object is at least 2 cm from edge • ... 	Transported out of service area. Service area has infinite height.	Standard criteria for valid placing: <ul style="list-style-type: none"> • object touches surface of service area • object does not move • ...
WORK PPT ^b	Same criteria with WORK BMT.	Same criteria with WORK BMT.	Object falls through the correct cavity and touches the ground beneath.
WORK RTT ^c	Minimum distance from edge of service area is 2 cm.	Same criteria with WORK BMT. Also: robot is not allowed to stop the object with its gripper.	Object is on the robot (implicit)
HEART TOGF ^d	Object is on the flat surface (implicit).	Robot holds the object for a minimum duration. Grasp's position and orientation matches task specification.	Not evaluated.
ACRV ^d	Arm base's centre is 1.5 m away from the shelf, rotated by $\sim 45^\circ$.	Object is in work order.	Object is placed in the tote without being drop from higher than 35 cm.

^a Basic Manipulation Test.

^b Precision Placement Test: robot must drop objects through corresponding cavities.

^c Rotating Table Test: robot must pick objects from a rotating table and place the object on itself.

^d Task-Oriented Grasping Functionality: robot must pick up the object based on some constraint, then put it back on the surface.

in the form of a Feature Model. We then argue how these *features* can be used to specify robotic requirements using the Behaviour-Driven Development (BDD) formulation. We also discuss the challenges of using BDD, even with extensions introduced in Gherkin, to formulate robotic scenarios as well as to make them executable. This motivates an approach to the BDD methodology which can accommodate the various domain knowledge involved in the scenarios, their interdependent relations, as well as different compositions of the scenarios themselves. Such an approach can be the basis for introducing a community-driven repository of BDD specifications for robotic competitions, similar to one proposed by [24]. Finally, automation of such an approach would still require a mechanism to communicate relevant runtime knowledge about the system, which informs how and when the specified acceptance criteria can be verified.

REFERENCES

- [1] F. Amigoni *et al.*, "Competitions for Benchmarking: Task and Functionality Scoring Complete Performance Assessment," *IEEE Robot. Automat. Mag.*, vol. 22, no. 3, pp. 53–61, Sep. 2015.
- [2] G. K. Kraetzschmar *et al.*, "RoboCup@Work: Competing for the Factory of the Future," in *RoboCup 2014: Robot World Cup XVIII*, ser. Lecture Notes in Computer Science, R. A. C. Bianchi *et al.*, Eds., vol. 8992, 2015, pp. 171–182.
- [3] N. Hochgeschwender *et al.*, *D3.1: HEART-MET Evaluation Plan*, Jun. 30, 2020.
- [4] J. Hart *et al.*, *RoboCup@Home 2022: Rules and regulations*, Jun. 30, 2022.
- [5] K. Kang *et al.*, "Feature-oriented domain analysis (FODA) feasibility study," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-90-TR-021, 1990.
- [6] D. North, "Behavior Modification: The evolution of behavior-driven development," *Better Software*, vol. 2006, no. 03, pp. 26–31, Jun. 5, 2006.
- [7] Y. Sun *et al.*, "Robotic Grasping and Manipulation Competition: Task Pool," in *Robotic Grasping and Manipulation*, ser. Communications in Computer and Information Science, Y. Sun and J. Falco, Eds., vol. 816, 2018, pp. 1–18.
- [8] Y. Sun *et al.*, "Research Challenges and Progress in Robotic Grasping and Manipulation Competitions," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 874–881, Apr. 2022.
- [9] H. Mnyusiwalla *et al.*, "A Bin-Picking Benchmark for Systematic Evaluation of Robotic Pick-and-Place Systems," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1389–1396, Apr. 2020.
- [10] J. Leitner *et al.*, "The ACRV picking benchmark to foster reproducible research," in *2017 IEEE Int. Conf. Robot. Autom. ICRA*, May 2017, pp. 4705–4712.
- [11] D. Bozhinoski *et al.*, "Safety for mobile robotic systems: A systematic mapping study from a software engineering perspective," *Journal of Systems and Software*, vol. 151, pp. 150–179, May 2019.
- [12] L. Brancalião *et al.*, "Systematic Mapping Literature Review of Mobile Robotics Competitions," *Sensors*, vol. 22, no. 6, p. 2160, Mar. 10, 2022.
- [13] S. Garcia *et al.*, "Software variability in service robotics," *Empir Software Eng.*, vol. 28, no. 2, p. 24, Mar. 2023.
- [14] A. Ortega, N. Hochgeschwender, and T. Berger, "Testing Service Robots in the Field: An Experience Report," in *2022 IEEE/RSJ Int. Conf. Intell. Robots Syst. IROS*, Oct. 23, 2022, pp. 165–172.
- [15] J. Guiochet, M. Machin, and H. Waeselync, "Safety-critical advanced robots: A survey," *Robotics and Autonomous Systems*, vol. 94, pp. 43–52, Aug. 2017.
- [16] J. Saenz *et al.*, "An online toolkit for applications featuring collaborative robots across different domains," *IEEE Transactions on Human-Machine Systems*, pp. 1–11, 2022. DOI: 10.1109/THMS.2022.3213416.
- [17] A. Nordmann *et al.*, "A Survey on Domain-specific Modeling and Languages in Robotics," *JOSE*, vol. 7, pp. 75–99, 2016.
- [18] S. Knoop, M. Pardowitz, and R. Dillmann, "Automatic robot programming from learned abstract task knowledge," in *2007 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2007, pp. 1651–1657.
- [19] T. Olsson, S. Sentilles, and E. Papatheocharous, "A systematic literature review of empirical research on quality requirements," *Requirements Eng.*, vol. 27, no. 2, pp. 249–271, Jun. 2022.
- [20] K. Beck, *Test-Driven Development: By Example* (The Addison-Wesley Signature Series). 2003, 220 pp.
- [21] X. Zheng *et al.*, "BraceAssertion: Runtime Verification of Cyber-Physical Systems," in *2015 IEEE 12th Int. Conf. Mob. Ad Hoc Sens. Syst.*, Oct. 2015, pp. 298–306.
- [22] M. Sirjani *et al.*, "Towards a Verification-Driven Iterative Development of Software for Safety-Critical Cyber-Physical Systems," *J Internet Serv Appl*, vol. 12, no. 1, p. 2, Dec. 2021.
- [23] Y. Deng *et al.*, "BMT: Behavior Driven Development-based Metamorphic Testing for Autonomous Driving Models," in *2021 IEEE/ACM 6th Int. Workshop Metamorph. Test. MET*, Jun. 2021, pp. 32–36.
- [24] M. Askarpour *et al.*, "RoboMAX: Robotic Mission Adaptation eXemplars," in *2021 Int. Symp. Softw. Eng. Adapt. Self-Manag. Syst. SEAMS*, May 2021, pp. 245–251.