

面向形式化多模态需求的领域特定语言

Marcos Gomez-Vazquez

Luxembourg Institute of Science and Technology

Esch-sur-Alzette, Luxembourg

0000-0001-7176-0793

Jordi Cabot

Luxembourg Institute of Science and Technology

University of Luxembourg

Esch-sur-Alzette, Luxembourg

0000-0003-2418-2489

摘要—多模态系统，处理多种输入类型如文本、音频和图像，在软件系统中变得越来越普遍，这得益于机器学习的巨大进步。这引发了对易于定义与这些新的用户交互类型相关的需求的需要，可能同时涉及一种以上的模态。由于缺乏适应多模态交互多样性的语言和方法，这仍然是一个开放性挑战，存在实现增强 AI 系统但未能充分满足用户需求的风险。

在这种意义上，本文提出了 MERLAN，一种领域特定语言 (DSL)，用于指定这些新型多模态界面的需求。我们呈现了这种语言的元模型以及作为 ANTLR 语法实现的文本语法。还提供了原型工具，使需求工程师能够编写此类需求，并自动生成与之兼容的系统的一种可能实现，基于代理框架之上。

Index Terms—领域特定语言，需求工程，多模态用户界面，代理

I. 介绍

随着机器学习和其他人工智能技术的爆炸式增长，软件系统正在迅速采用需要处理文本、音频和图像等新输入模态的新类型复杂用户界面。有时，同一时间会使用多种类型的模态。这种类型的界面被称为多模态用户界面 (MUI)。MUI 提供了一个具备人与人人机接口功能的界面 [1]。这是人类计算机交互 (HCI) 领域内的一个广泛领域，其应用和定义根据上下文的不同而有所不同。术语“模态”指的是在人与计算机之间用作输入和输出的通信模式，如文本、音频和图像，尽管在其他情况下会考虑其他模态（例如，面部表情、手势等）。

尽管由于新多模态大型语言模型 (LLMs) 等不断涌入的技术，使得构建这种增强型 AI 系统变得更加容易，这些技术促进了对多模态输入的分析，但验证该系统是否满足用户实际需求正变得越来越复杂。确实，我们缺乏适当的软件需求工程语言和技术来促进精确指

定应触发系统响应的 MUI 条件、从多模态输入中收集以提供适当响应的数据（在 MUI 术语中的“实体”）以及实际响应 [2]。

对于后者，当前的语言可以重复使用，但对于前两个元素，则需要一种新的指定由 MUI 驱动的需求的方法。该领域的初步工作集中在聊天机器人的需求上。特别是，为了识别出用户“意图”（即用户的意愿或目标）以及哪些实体或参数是与特定意图 [3] 相匹配的用户文本输入的一部分。然而，对于其他模式（如图像或音频）以及其他涉及多个输入的需求，并非如此。

为了解决这个问题，本文提出了 MERLAN：一种多模态环境需求语言¹。MERLAN 旨在标准化多模态需求的定义，独立于用于处理多模态数据的基础技术。MERLAN 是一种领域特定语言 (DSL)，通过语言元模型进行形式化，并支持文本具体语法，以帮助用户轻松定义 MUI 需求。还提供了工具支持。它包括语言支持，以及一个将需求转换为 Python 代理的概念验证实现，当满足需求条件时，该代理能够触发系统响应。

本文其余部分结构如下：第 II 节引入了一个运行示例以激发并说明我们的领域特定语言 (DSL)；第 III 节介绍了 DSL 的抽象语法，而第 IV 节则专注于其具体的语法；第 V 节展示了提供的工具支持；第 VI 节讨论了本论文的相关工作；研究路线图在第 VII 节中进行讨论，最后，第 VIII 节以结论结束。

II. 运行示例

为了激励和说明我们的方法，本节介绍一个将在整篇论文中使用的示例。

This project is supported by the Luxembourg National Research Fund (FNR) PEARL program, grant agreement 16544475.

¹<https://github.com/BESSER-PEARL/merlan>

让我们设想我们需要指定一个新的家庭自动化和安全系统的要求，以下简称房屋代理。该房屋代理具有一些输入设备来捕捉文本、声音、视频、温度、光线和运动。它还具有用于文本和音频的输出设备，以及执行一组预定义动作的能力（即，如拨打电话或触发警报等操作）。

不论负责从现实世界捕获信息的实际软件组件（例如，用于检测来自摄像头输入的对象 ML 模型），房屋代理都需要提供明确定义的实体，这些实体可以从所有不同的输入设备中识别出来，并且需要有描述如何评估这些实体以便决定是否触发特定响应的规则。

例如，对于图像输入设备，我们可以定义具体的实体如 `person`, `dog`, `car`, `smoke` 和 `fire`。一个音频实体可以是 `strong_sound`。此外，这些实体可以具有属性。`person` 实体可以拥有 `gender` 和 `ethnicity` 属性，在代理识别过程中，这些属性应被填入正确的值。系统需要定义其他类型的更为抽象的实体，例如“白天或夜晚”或“空房子”。它们更加抽象，因为必须从输入数据的部分推断出这些内容，而不是直接将其识别为其中的对象。

定义了所有这些实体后，我们可以设计代理将检查以触发某些操作的规则。例如：

- 1) 如果检测到烟雾：通知房屋主人。
- 2) 如果（检测到火情）或（房屋无人且（检测到一些车辆或人员））：激活警报，通知房主并报警。
- 3) 如果（声音很大）且（在夜间）：打开灯光
- 4) 如果检测到带有未识别车牌的车辆：通知房主

第一条规则简单地定义了一个具体实体（烟雾）的存在。第二条由多个具体的和抽象的要求通过“和”与“或”操作符组成。第三条期望检测到一个音频的具体实体（强音）和一个图像的抽象实体（夜晚）。最后一个要求在不仅识别了一个实体（汽车），而且还识别了特定属性值（车牌）的情况下得到满足。

这一组规则和实体将组成房屋代理多模态接口的需求，描述系统应基于哪些数据对什么条件作出反应以及在需求满足时应采取的行动。这些需求可以使用自然语言来描述，但精确度不足以有信心地实现它们。

因此，我们认为需要一种领域特定语言来以更精确的方式正式定义 MUI 需求。下一节将介绍我们为此提出的 MERLAN 领域特定语言。

III. DSL 设计

领域特定语言 (DSL) 是一种专门设计用于解决特定领域问题的编程或建模语言。与通用语言不同，DSL 提供了针对该领域的概念和规则量身定制的更高层次抽象，使表达模型、转换和约束 [4] 变得更容易。DSL 由定义其主要概念（及其相互关系）的抽象语法和实现它的具体语法组成（通常通过文本或图形表示）。

DSL 的抽象语法通过与任何具体的语法或符号无关的元模型规范定义其结构表示。本节介绍了 MERLAN 核心抽象语法的元模型，用于表达 MUI 需求，概述了其关键元素和关系。

图 1 使用 UML 类图形式表达了这个元模型，如同往常一样。

下一节将更详细地描述元模型的主要元素。我们将解释分为两个小节，一节涵盖专注于多模态需求定义的元类，另一节则更多关注那些需求中引用的实体定义。

A. 多模态需求的定义

`AMultimodalRequirement` 定义了在多模态输入系统中需要评估的规则。这些规则提供了系统应匹配以触发某些动作的条件形式化定义。

元模型定义了两种类型的需求：简单和复杂。

1) 复杂需求：`ComplexRequirement` 是 `MultimodalRequirements` 的组合，其本身可以是复杂的也可以是简单的。需求通过布尔运算符组成，即 `AND`（其下的所有需求必须满足）、`OR`（至少有一个需求必须满足）和 `NOT`（其下的需求条件不能被满足）。

2) 简单要求：`ASimpleRequirement` 表达了对单个实体的一个条件。当定义一个简单的需求时，我们创建了一条当检测到实体时将被满足的规则。简单需求有一个 `confidence` 属性来指示为了满足该需求所需达到的最低置信度（即对实体检测的置信度）。

`SimpleRequirement` 包含 `AbstractRequirement` 和 `ConcreteRequirement` 来区分引用具体和抽象实体的简单需求。它们之间的区别在于，一个 `ConcreteRequirement` 可以定义基数来指定应检测到的该实体实例的数量。基数符号是基于 UML 基数的，允许定义特定值 (`[n]`，恰好 n 个实例)、具体区间 (`[m..n]`，其中 $m < n$ 和 $m \geq 0$) 或无界区间 (`[n..*]`，其中 $n \geq 0$)。正如我们稍后将看到的，实体可以具有属性，这些属性也可以在匹配过程中使用。一

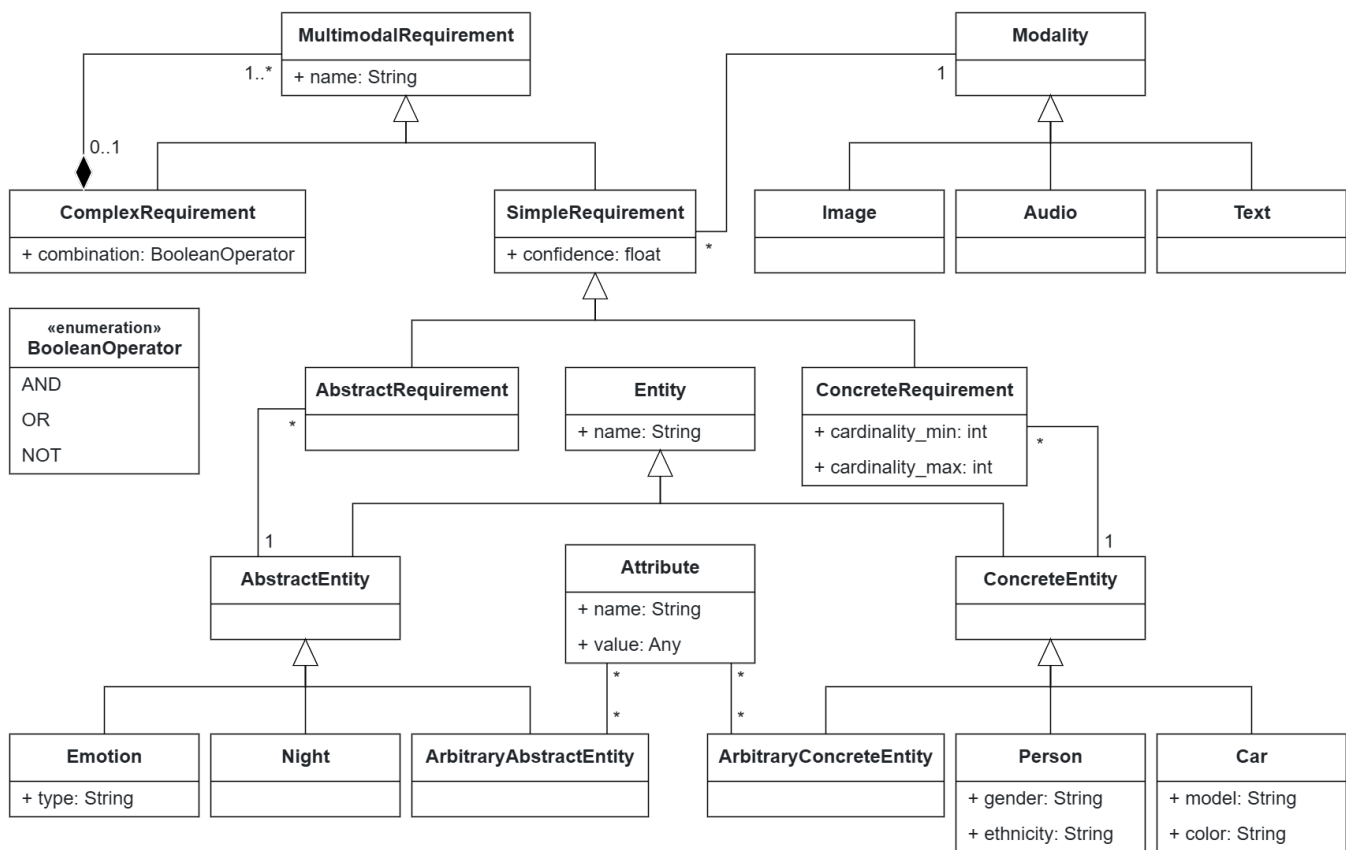


图 1. MERLAN 元模型

个SimpleRequirement 具有特定的模式，指示应考虑哪种模式以满足要求。例如，一个引用具体实体Person 并具有模态Image 的SimpleRequirement 将基于图像输入进行评估（即，在图像中寻找一个人），而相同要求但具有模态Audio 的则会分析音频输入以找到一个人（即，如果有人说话且语音被检测到）。在评估需求时，某些属性仅对特定模态有意义（例如，实体Car 的color 仅对图像模态有意义）。这在定义实体属性时需要考虑。

B. 实体的定义

一个Entity 表示现实世界中的概念，无论是物质的还是抽象的，都可以在某种多模态输入中被明确识别。定义实体最简单的方法是通过名称和属性来指定其结构和内容。当定义一个实体时，我们定义它的属性，这些属性可能具有特定值或为空值（即未知值）。

设置属性值强制识别引擎识别那些具有该确切属性值的实体。例如，如果我们定义一个带有gender = "male"属性的Person 实体，我们可以在需

求中使用该实体以将其适用范围限制为男性。

属性值可以留空，目的是在识别过程中动态推断其值。以实体Person 为例，一个值为空的gender 属性应使识别引擎识别男性和女性，并在运行时填充所识别Person 的gender 值。空值属性还充当占位符，在需求级别设置这些占位符。这种机制通过允许定义通用实体并将特定上下文细节推迟到个别需求，提供了灵活性。

我们根据实体的性质考虑两种类型的实体，即抽象实体和具体实体。

1) 具体实体：A ConcreteEntity 是Entity 的一个子类，表示在多模态环境中可以描述其物理存在并可以在该环境中引用或与其互动的具体对象或实体。

2) 抽象实体：AbstractEntity 是Entity 的一个子类，它没有明确限定的物理存在，但代表可以从环境中推断出的概念、想法、属性或分类。一个抽象实体的例子可能是night，表示图像是否在夜间拍摄。更抽象的一个例子可能是hazard，表示从图像或音频中感知到的危害程度（我们可以为此属性定义一些属性来描述我

们的“危害标准”)。

3) 预定义和任意实体：元模型图包括具体实体和抽象实体的示例实例及其默认属性。例如，一个简化的 **Person** 具体实体可以根据属性 **gender** 和 **ethnicity** 来定义。抽象和具体实体有一个 **ArbitraryEntity** 子类，允许定义具有任意属性的自定义或领域特定实体。

为了简化给定领域中需求的定义，我们可以基于该领域的现有本体预先定义最相关的实体。这是领域特定语言大小与其易用性之间的权衡。

IV. 具体语法

本节介绍了我们 DSL 的具体语法，该语法符合之前详细描述的元素模型。具体语法定义了确保语言使用一致和标准化的规则。在本文中，我们介绍了一种使用 ANTLR 实现的文本具体语法，ANTLR 是一种强大的工具，可以从语法规则生成语言解析器。

我们首先介绍语法，然后给出一个遵循该语法的 MERLAN 规范示例。列表 1 展示了 DSL 语法的一个摘录。

Listing 1. MERLAN ANTLR grammar; main rules.

```
1 grammar MERLAN;
2
3 script
4   : entities?
5     requirements?
6   ;
7
8 entities
9   : ENTITIES NEWLINE
10    concrete_entities?
11    abstract_entities?
12  ;
13
14 concrete_entities
15   : CONCRETE NEWLINE concrete_entity*
16  ;
17
18 concrete_entity
19   : ID NEWLINE attribute*
20  ;
21
22 // Other rules omitted for brevity
23
24 requirement
```

```
25   : complex_requirement
26     | simple_requirement
27   ;
28
29 complex_requirement
30   : (AND | OR) NEWLINE requirement+
31     | NOT NEWLINE requirement
32   ;
33
34 simple_requirement
35   : abstract_requirement
36     | concrete_requirement
37   ;
38
39 concrete_requirement
40   : CONCRETE cardinality? NEWLINE attribute*
41   ;
42
43 // Other rules omitted for brevity
```

语法定义了 MERLAN 语言中允许的语法规则。在第一级，**script** 规则表明可以定义 **entities** 和 **requirements**。

一些规则以大写关键字开头，将代码划分为清晰的部分（例如，参见 **ENTITIES**）。**entities** 部分可以包含具体或抽象的实体。为了简洁起见，本文中省略了 **abstract_entities** 规则（以及其他一些规则），但它与 **concrete_entities** 类似。一个具体的实体由其名称（即 ID）和任意数量的属性定义。

requirement 可以是复杂的，也可以是简单的。**complex_requirement** 以一个布尔运算符（AND、OR 或 NOT）开始，并且有一组要求（如果运算符是 NOT，则限于 1）。一个 **simple_requirement** 可以是抽象的也可以是具体的，其中 **concrete_requirement** 可以具有自定义基数。简单需求包含一组属性，其中一些是强制性的。强制性属性未指定，因为语法的目的在于确保句法一致性。在解析器执行期间（即解析 MERLAN 代码脚本时）对缺失属性（以及基数正确性）进行评估。强制性属性包括 **confidence**、**modality** 和 **entity**（这些仅适用于简单需求）以及 **name**。

列表 2 显示了一个示例 MERLAN 代码，该代码定义了实体和需求，遵循之前提出的房屋代理运行示例，如第 II 节所示。

Listing 2. Example MERLAN code.

```

1 ENTITIES:
2   CONCRETE:
3     person
4       - gender: ?
5       - ethnicity: ?
6     smoke
7     fire
8     dog
9       - breed: "labrador"
10    car:
11      - model: ?
12      - color: ?
13  ABSTRACT:
14    night
15      - description: "The image is taken at
16                      night"
17    empty_house
18      - description: "The house is empty"
19 REQUIREMENTS:
20   requirement1:
21     CONCRETE
22       - entity: smoke
23       - name: "smoke"
24       - modality: "image"
25       - confidence: 0.5
26   requirement2
27     OR
28     CONCRETE
29       - entity: fire
30       - name: "fire"
31       - confidence: 0.5
32       - modality: "image"
33     AND
34     ABSTRACT
35       - entity: empty_house
36       - name: "empty_house"
37       - confidence: 0.3
38       - modality: "image"
39     OR
40     CONCRETE [1..*]
41       - entity: person
42       - name: "unknown_person"
43       - confidence: 0.7
44       - modality: "image"
45       - gender: "male"
46     CONCRETE [1..*]
47       - entity: car

```

```

47       - name: "unknown_car"
48       - confidence: 0.7
49       - modality: "image"

```

第一个代码块，标识符为ENTITIES，包含所有遵循语法规则的实体定义。在此示例中，存在没有属性的实体（参见smoke和fire），一个具有特定值属性的实体dog（breed: "labrador"）以及其他具有空值属性的实体（person和car）。

在REQUIREMENTS关键字下的需求块包含第II节中定义的2个示例需求。requirement2包含一组复杂的需求，其中两个内部简单需求定义了[1..*]的基数（即最小实例数为1）。引用person实体的需求展示了如何在需求级别设置实体属性的值（参见gender: "male"属性）。

V. 工具支持

完整的ANTLR语法以使用MERLAN可在我们的GitHub开源仓库中获得。

通过该语法，需求工程师可以描述完整的MERLAN规范。但为了使这些规范更具操作性，并从它们中获得更好的投资回报(ROI)，我们还实现了一个概念验证实施，以推导出一个代理实现，该实能够理解和响应需求条件，有效触发基于检测到的多模型输入条件下满足情况的状态变化。

更具体地说，我们实现了一种转换，该转换给定一个MERLAN规范，会创建一个基于BESSER代理框架²的代理，其中MERLAN需求被转化为一组触发条件。列表3显示了示例MERLAN代码在列表2中生成的代码

变换是通过自定义实现的树访问者模式来完成的，该模式遍历由MERLANANTLR解析器生成的抽象语法树。请注意，这种方法可以很容易地复制到支持类似基于需求的转换的任何语言或代理框架中。

我们选择的代理框架集成了大型语言模型和计算机视觉模型，从多模态环境中捕获信息，并实时用于评估需求匹配。生成的代码包括与代理实体数据库关联的多模态实体以及需求条件本身。基于此输入处理组件，代理开发者可以完成代理规范的制定，表达在触发条件下的响应行为。

²<https://github.com/BESSER-PEARL/BESSER-Agentic-Framework>

Listing 3. Generated Python code.

```

1 # Entities
2 person = ConcreteEntity(name="person", attributes={"gender": None, "ethnicity": None})
3 smoke = ConcreteEntity(name="smoke", attributes={})
4 fire = ConcreteEntity(name="fire", attributes={})
5 dog = ConcreteEntity(name="dog", attributes={"breed": "labrador"})
6 car = ConcreteEntity(name="car", attributes={"brand": None, "model": None, "color": None})
7 night = AbstractEntity(name="night", attributes={"description": "The image is taken at
    night"})
8 empty_house = AbstractEntity(name="empty_house", attributes={"description": "The house is
    empty"})
9 # Requirements
10 requirement1 = RequirementDefinition("requirement1")
11 requirement1.set(ConcreteRequirement(name="smoke", concrete_entity=smoke,
    attributes={"modality": "image", "confidence": 0.5}))
12 requirement2 = RequirementDefinition("requirement2")
13 requirement2.set(
14     OR([
15         ConcreteRequirement(name="fire", concrete_entity=fire, attributes={"confidence": 0.5,
            "modality": "image"}),
16         AND([
17             AbstractRequirement(name="empty_house", abstract_entity=empty_house,
                attributes={"confidence": 0.3, "modality": "image"}),
18             OR([
19                 ConcreteRequirement(name="man", concrete_entity=person, attributes={"min": 1, "max":
                    0, "confidence": 0.7, "modality": "image", "gender": "male"}),
20                 ConcreteRequirement(name="unknown_car", concrete_entity=car, attributes={"min": 1,
                    "max": 0, "confidence": 0.7, "modality": "image"})
21             ])
22         ])
23     ])
24 )
25
26 # The following code is not automatically generated. It illustrates how to use a requirement
    to define transitions between the agent states (agent definition code omitted for brevity)
27 initial_state.when_requirement_matched_go_to(requirement1, smoke_state)

```

VI. 相关工作

在本节中，我们讨论与 MUI 需求规范相关的研究工作。首先关注明确涉及创建 MUI 的框架，然后涵盖其他更接近需求工程领域的方法，提出用于表达某些类型高级界面需求的语言。

A. 多模态用户界面

最近关于 MUIs 的工作集中在利用机器学习技术从数据 [5] 中自动提取模式和见解。

为了这个目的，已经提出了多种框架。Xspeak [6] 在 X Window System 上添加了一个语音接口，允许使用词语与窗口进行交互。开放代理架构 (OAA) [7] 提出了一个包含口语语言、手写和手势的多代理系统的框架。Openinterface [8] 是另一个用于设计 MUIs 的工具，

提供了自己的运行时环境和集成开发环境。Squidy 交互库 [9] 提出了一种库以减少设计 MUI 的工作量，在一个共同的库中整合了不同的工具包和框架，通过提供一种可视化语言和技术设备集合来隐藏复杂性。

在机器人和医疗保健领域，对多模态用户界面 (MUIs) 也有特别的兴趣。例如，AMIR [10] 是一个具有语音和手势基础接口的辅助机器人，而 FIRMA [11] 则是一个为老年人友好互动多模态应用开发框架的辅助机器人。多模态用户界面在智能家居用户界面 [12] 中也有很强的存在感。

然而，所有这些方法都集中在 MUIs 本身的发展上，而不是它们应该实现的需求的形式化，这正是我们 MERLAN 提案的目的。

B. 领域特定语言

DSLs 在许多机器学习问题 [13] 中使用，包括专注于形式化需求的 DSLs [14]。一个示例是 Impromptu [15] 提出了一种 DSL 以模块化和工具无关的方式定义结构化的提示。其他 DSL 更加关注机器学习中的公平性方面，例如 [16], [17]。

更接近我们的工作，其他领域特定语言涵盖了软件组件的多模态方面。例如，SEMKIS [18] 关注数据集和神经网络的需求工程以提高识别能力。ViSaL [19] 允许程序员表达图像质量检测规则来强制执行安全约束，增加机器人感知系统的可信度。FVision [20] 作为 Haskell 库设计为领域特定语言来构建和测试视觉跟踪系统。Midgar IoT 平台通过添加计算机视觉模块扩展以自动化摄像头输入分析。他们的方法仅检测人员，其他物体需要模型训练。他们提出开发一种 DSL 来简化计算机视觉流水线 [21]。请注意，存在可用于评估此类系统的现有资源，例如 CLEVR [22]，它提出了一个基准数据集用于智能系统中的视觉推理。更具体地针对聊天机器人领域，[3] 和 [23] 提出了文本输入中意图匹配和实体识别的 DSL。

请注意，上述示例针对特定类型的输入，并且其中许多是针对特定环境的，而 MERLAN 旨在提供一种结合不同类型的需求和实体（包括具体和抽象的）及其条件的多模态需求解决方案，为创建能够满足用户需求的强大多模态代理打开了大门。

VII. 研究路线图

为了全面定义一种强大且表达力丰富的面向多模态用户界面的领域特定需求语言，必须解决几个关键方面。以下是其中的一些：

- 增加图形表示到 MERLAN 中，增强其易用性，并使不太具备技术能力的从业者和工程师更容易指定和可视化 MUI 需求。这应包括通过示例进行视觉建模的组件，用户可以给出图像作为触发系统操作的示例场景。或者，也可以提供以自然语言描述需求的选择。机器学习可用于支持此任务。
- 扩展语言以涵盖在条件匹配时执行的行为需求规范。对于“传统”动作，现有的行为语言（例如 UML 规范）可能就足够了，但对于多模态响应，扩展 MERLAN 可能是更好的选择，在 MERLAN 中多模态已经是首要元素。
- 时间条件用于在需求中表达时间约束，例如，在触发某个动作之前视频中某对象的持续时间约束。
- 此外，应探索分层模式，其中高层级模式（例如手势或面部表情）由低层级模式（例如图像或视频）派生。这种分层结构将使复杂多模态交互的需求定义更加精确和灵活。
- 质量分析以检测多模态需求中的不一致性和冲突。例如，某些属性，如颜色，在基于图像的模态中才相关，并且可能在文本或听觉上下文中不适用。或者更难检测的是，在图像中的两个条件可能是互斥的，这意味着这样的条件永远无法满足。

VIII. 结论

在本文中，我们介绍了 MERLAN，一种旨在标准化多模态用户界面 (MUI) 需求规范的领域特定语言。我们的方法利用了基于元模型的形式化和文本具体语法，以促进规范过程，并提供了一个概念验证实现，展示了将这些需求转化为可执行代理代码的潜在路径，以便在实时多模态环境中执行这些需求。

未来的工作将重点放在通过解决上述讨论的开放挑战来扩展 MERLAN 的能力。此外，我们计划进行实证验证实验，并扩展我们的工具支持，以促进研究人员和该领域的实践者采用我们的语言和基础设施。作为此类工具扩展的一部分，我们打算开发一个基于现有本体论的预定义实体库，以便在表达新需求时导入和重用。

参考文献

- [1] M. Z. Baig and M. Kavakli, “Multimodal systems: Taxonomy, methods, and challenges,” 2020. [Online]. Available: <https://arxiv.org/abs/2006.03813>
- [2] L. M. Reeves, J. Lai, J. A. Larson, S. Oviatt, T. S. Balaji, S. Buisine, P. Collings, P. Cohen, B. Kraal, J.-C. Martin, M. McTear, T. Raman, K. M. Stanney, H. Su, and Q. Y. Wang, “Guidelines for multimodal user interface design,” *Commun. ACM*, vol. 47, no. 1, p. 57 – 59, Jan. 2004. [Online]. Available: <https://doi-org.proxy.bnl.lu/10.1145/962081.962106>
- [3] G. Daniel, J. Cabot, L. Deruelle, and M. Derras, “Xatkit: A multimodal low-code chatbot development framework,” *IEEE Access*, vol. 8, pp. 15 332–15 346, 2020.
- [4] M. Mernik, J. Heering, and A. M. Sloane, “When and how to develop domain-specific languages,” *ACM Comput. Surv.*, vol. 37, no. 4, p. 316 – 344, Dec. 2005. [Online]. Available: <https://doi-org.proxy.bnl.lu/10.1145/1118890.1118892>
- [5] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, “Multimodal machine learning: A survey and taxonomy,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 423–443, 2019.
- [6] C. Schmandt, M. Ackerman, and D. Hindus, “Augmenting a window system with speech input,” *Computer*, vol. 23, no. 8, pp. 50–56, 1990.
- [7] D. B. Moran, A. J. Cheyer, L. E. Julia, D. L. Martin, and S. Park, “Multimodal user interfaces in the open agent architecture,” in *Proceedings of the 2nd International Conference on Intelligent User Interfaces*, ser. IUI ’97. New York, NY, USA: Association for Computing Machinery, 1997, p. 61 – 68. [Online]. Available: <https://doi-org.proxy.bnl.lu/10.1145/238218.238290>
- [8] M. Serrano, L. Nigay, J.-Y. L. Lawson, A. Ramsay, R. Murray-Smith, and S. Denef, “The openinterface framework: a tool for multimodal interaction,” in *CHI ’08 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA ’08. New York, NY, USA: Association for Computing Machinery, 2008, p. 3501 – 3506. [Online]. Available: <https://doi.org/10.1145/1358628.1358881>
- [9] W. A. König, R. Rädle, and H. Reiterer, “Interactive design of multimodal user interfaces,” *Journal on Multimodal User Interfaces*, vol. 3, no. 3, pp. 197–213, Apr 2010. [Online]. Available: <https://doi.org/10.1007/s12193-010-0044-2>
- [10] D. Ryumin, I. Kagiroy, A. Axyonov, N. Pavlyuk, A. Saveliev, I. Kipyatkova, M. Zelezny, I. Mporas, and A. Karpov, “A multimodal user interface for an assistive robotic shopping cart,” *Electronics*, vol. 9, no. 12, 2020. [Online]. Available: <https://www.mdpi.com/2079-9292/9/12/2093>
- [11] N. Kazepis, M. Antona, and C. Stephanidis, “Firma: A development framework for elderly-friendly interactive multimodal applications for assistive robots,” 04 2016.
- [12] M. Blumendorf and S. Albayrak, “Towards a framework for the development of adaptive multimodal user interfaces for ambient assisted living environments,” in *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments*, C. Stephanidis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 150–159.
- [13] I. Portugal, P. Alencar, and D. Cowan, “A preliminary survey on domain-specific languages for machine learning in big data,” in *2016 IEEE International Conference on Software Science, Technology and Engineering (SWSTE)*, 2016, pp. 108–110.
- [14] Z. Pei, L. Liu, C. Wang, and J. Wang, “Requirements engineering for machine learning: A review and reflection,” in *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, 2022, pp. 166–175.
- [15] S. Morales, R. Clarisó, and J. Cabot, “Impromptu: a framework for model-driven prompt engineering,” *Software and Systems Modeling*, Jan 2025. [Online]. Available: <https://doi.org/10.1007/s10270-024-01235-4>
- [16] A. Yohannis and D. Kolovos, “Towards model-based bias mitigation in machine learning,” in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 143 – 153. [Online]. Available: <https://doi.org/10.1145/3550355.3552401>
- [17] J. Giner-Miguel, A. Gómez, and J. Cabot, “Describeml: A dataset description tool for machine learning,” *Sci. Comput. Program.*, vol. 231, p. 103030, 2024. [Online]. Available: <https://doi.org/10.1016/j.scico.2023.103030>
- [18] B. Jahić, N. Guelfi, and B. Ries, “Semkis-dsl: A domain-specific language to support requirements engineering of datasets and neural network recognition,” *Information*, vol. 14, no. 4, 2023. [Online]. Available: <https://www.mdpi.com/2078-2489/14/4/213>
- [19] J. T. M. Ingbergsson, D. Kraft, and U. P. Schultz, “Safety computer vision rules for improved sensor certification,” in *2017 First IEEE International Conference on Robotic Computing (IRC)*, 2017, pp. 89–92.
- [20] J. Peterson, P. Hudak, A. Reid, and G. Hager, “Fvision: A declarative language for visual tracking,” in *Practical Aspects of Declarative Languages*, I. V. Ramakrishnan, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 304–321.
- [21] C. González García, D. Meana-Llorián, B. C. Pelayo G-Bustelo, J. M. Cueva Lovelle, and N. Garcia-Fernandez, “Midgar: Detection of people through computer vision in the internet of things scenarios to improve the security in smart cities, smart towns, and smart homes,” *Future Generation Computer Systems*, vol. 76, pp. 301–313, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X16308652>
- [22] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, “Clevr: A diagnostic dataset for compositional language and elementary visual reasoning,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1988–1997.
- [23] S. Pérez-Soler, E. Guerra, and J. de Lara, “Model-driven chatbot development,” in *Conceptual Modeling - 39th International Conference, ER 2020, Vienna, Austria, November 3-6, 2020, Proceedings*, ser. Lecture Notes in Computer Science, G. Dobbie, U. Frank, G. Kappel, S. W. Liddle, and H. C. Mayr, Eds., vol. 12400. Springer, 2020, pp. 207–222. [Online]. Available: https://doi.org/10.1007/978-3-030-62522-1_15