Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный индустриальный университет»

Посвящается 100-летию со дня рождения ректора СМИ, доктора технических наук, профессора Н.В.Толстогузова

НАУКА И МОЛОДЕЖЬ: ПРОБЛЕМЫ, ПОИСКИ, РЕШЕНИЯ

ГУМАНИТАРНЫЕ НАУКИ

ВЫПУСК 25

Труды Всероссийской научной конференции студентов, аспирантов и молодых ученых 12 – 14 мая 2021 г.

ЧАСТЬ III

Под общей редакцией профессора Н.А. Козырева

Новокузнецк
2021

ББК 74.48.278
Н 340

Н 340 Наука и молодежь: проблемы, поиски, решения: труды Всероссийской научной конференции студентов, аспирантов и молодых ученых, 12–14 мая 2021 г. Выпуск 25. Часть III. Гуманитарные науки / Министерство науки и высшего образования Российской Федерации, Сибирский государственный индустриальный университет ; под общ. ред. Н.А. Козырева – Новокузнецк; Издательский центр СибГИУ, 2021. – 452 с. : ил.

ISSN 2500-3364

Представлены труды Всероссийской научной конференции студентов, аспирантов и молодых ученых по результатам научно-исследовательских работ. Третья часть сборника посвящена актуальным вопросам иностранного языка, образования, культуры, социально-гуманитарных дисциплин, спорта, здоровья.

Материалы сборника представляют интерес для научных и научно-технических работников, преподавателей, аспирантов и студентов вузов.

ISSN 2500-3364

В этой статье мы рассмотрели проблему оценки сложности предложений на итальянском языке, и влияние схемы представления на результаты модели.

Мы использовали одну и ту же нейронную модель для решения задачи классификации, и мы пробовали разные схемы представления. Результаты наших тестов показывают, что значение показателей, определяющих правильность модели, имеют малое отличие среди всех методов представления. Это позволяет предположить, что проблема оценки сложности текста больше связана с характером модели, чем с методом представления признаков предложения.

Библиографический список

1. Evaluation of Text Complexity in Italian Language: a Representation Point of View // ScienceDirect URL: https://doi.org/10.1016/j.procs.2018.11.108 (дата обращения: 20.10.2019).

УДК 811.111:004.43

ФУНКЦИОНАЛЬНАЯ МОДЕЛЬ АКТОРНОГО, АГЕНТСКОГО, ФУНКЦИОНАЛЬНОГО, ОБЪЕКТНОГО И ПРОЦЕДУРНОГО ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Терехов Д.А.
Научный руководитель: канд. пед. наук, доцент Моисеенко Т.Г.

Сибирский государственный индустриальный университет,
г. Новокузнецк, e-mail: dima00-10@mail.ru

В этой статье предпринята попытка идентифицировать, определить и упорядочить основные концепции, лежащие в основе стилей программирования актора, агента, функционала, объекта и процедуры, как они реализованы в практических языках программирования.

Ключевые слова: языки программирования; языковые конструкции программирования; агентно-ориентированное программирование; функциональное программирование; объектно-ориентированное программирование.

Функциональное моделирование поддерживает неформальное сравнение существующих и будущих систем, характеризуя системы и их функции как примеры концепций предметной области. Апель и Кестнер идентифицируют десять различных определений термина «функциональный», отражая тот факт, что функциональное моделирование может применяться на многих этапах жизненного цикла программного обеспечения и на уровнях детализации, начиная от анализа предметной области и заканчивая настройкой операционных систем во время компиляции.

Функциональное моделирование обычно используется для управления изменчивостью в контексте линейки программных продуктов. Однако основное внимание в этом документе уделяется функционально-ориентированному анализу предметной области языков прикладного программирования высокого уровня с целью определения «функций и возможностей класса связанных программных систем». Функциональное моделирование - это творческое занятие, которое часто является итерактивным и ориентированным на сообщество.

Сравнения языков прикладного программирования на основе признаков включают в себя множество идей из более ранних классификаций и таксономий, с дополнительным акцентом на оптимизацию моделей, чтобы максимизировать компоновку, уменьшить зависимости между функциями и, таким образом, минимизировать взаимодействия функций. Сравнительные исследования, основанные на характеристиках и основах, имеют несколько ключевых характеристик: главная цель обоих типов исследований – объединить выбранную работу в рамках заранее определенной границы для создания единой связной модели. В отличие от обзоров, которые стремятся быть всесторонними, сравнения на основе структуры и признаков обычно фокусируются на концепциях более высокого уровня и взаимосвязях между ними.

Абстрактная модель семейства продуктов, такая как функциональная модель, может оцениваться либо путем изучения одного экземпляра продукта в его предполагаемом контексте, либо путем анализа подмножества экземпляров продукта относительно модели. В данной статье принят последний подход; каждый экземпляр продукта является хорошо известным языком программирования.

Следует подчеркнуть, что представленная модель признаков не является ни окончательной, ни безусловной; она может и должна быть изменена и расширена, чтобы приспособить новые языки и события.

Модель объектов принимает форму дерева с девятью узлами первого уровня. Каждый узел представляет либо элемент, либо группу элементов - группу связанных или концептуально похожих элементов.

1. Система типов

Системы типов служат трем связанным целям в языках программирования: классифицировать значения, определять их применимые операции и сообщать компилятору, сколько памяти выделить для хранения значения данного типа. На первый взгляд, тип - это «ограничение, которое определяет набор допустимых значений, которые ему соответствуют». Также типы являются абстрактными «спецификациями функциональности», которые определяют «законные контексты использования для значений, которые они описывают».

Попытка выполнить недопустимую операцию со значением известна как ошибка типа, которая может быть обнаружена либо во время компиляции, либо во время выполнения. В языке без типов «должно быть так, что каждое значение может использоваться в любом контексте».

2. Неизменное состояние

Важной особенностью многих языков программирования является «возможность ассоциирования значений с символами». Эта категория объектов связана с символьно-ценностными ассоциациями, которые после создания не могут быть изменены.

3. Изменчивое состояние

Ван Рой и Хариди определяют именованное состояние вычислительной сущности как «последовательность значений во времени, которая содержит промежуточные результаты желаемого вычисления». Хотя не все языки программирования обеспечивают явное представление состояния, любой объект, который знает свое прошлое, должен хранить эти знания либо внутри, либо снаружи в среде.

4. Декларативные выражения

Финкель и Камин определяют декларативное программирование как разделение логики и контроля. Программист создает логический компонент, состоящий из декларативных выражений, которые «определяют, каким должен быть результат алгоритма». Компонент управления частично или полностью предоставляется компилятором или интерпретатором.

5. Императивный контроль

Императивный контроль позволяет программисту явно указывать порядок выполнения или оценки операторов или выражений во времени. Определенные программистом последовательности операторов являются центральными для императивных языков, а также должны поддерживать структурированный ввод / вывод в декларативных языках.

6. Явный параллелизм

Два действия являются одновременными, если они могут чередоваться в любом порядке или выполняться параллельно, как это определено базовой платформой в соответствии с количеством доступных ядер ЦП. Хотя декларативные программы часто могут распараллеливаться автоматически компилятором или интерпретатором, многие языки также определяют функции для явной поддержки одновременного выполнения программ, которые зависят от изменчивого состояния или императивного управления. Взаимодействие между явно параллельными действиями может поддерживаться в нескольких различных стилях связи.

7. Модульность

Особенности модульности позволяют разделить систему на «согласованные части, которые можно разрабатывать и обслуживать отдельно», а затем, при необходимости, повторно использовать как внутри компании, так и за ее пределами, для получения экономической выгоды. Болдуин и Кларк предлагают общую теорию модульности, основанную на шести операторах, которые кратко описывают возможные пути эволюции для модульной структуры. Для программных систем две из них - расщепление и подстановка - требуют явной поддержки на уровне языка программирования.

8. Метапрограммирование.

Метапрограммы анализируют, модифицируют и генерируют программы. Поскольку построение компилятора и статический анализ выходят за рамки данной статьи, эта категория функций конкретно касается рефлексивного метапрограммирования во время выполнения программ, которые анализируют, модифицируют и расширяют себя. Функции метапрограммирования классифицируются здесь в соответствии с типами программных артефактов, с которыми они работают.

9. Ввод, вывод.

Эта категория функций связана с вводом и выводом данных о пользователях и «средах». Среда обеспечивает доступ к ресурсам (как аппаратным, так и программным) и обеспечивает «условия», при которых существуют действующие лица, агенты или объекты. Среда может состоять из множества одновременных действий, но в отличие от одновременных действий, среда часто определяется на другом (обычно низкоуровневом) языке программирования.

Таким образом, в этой статье предлагается и проверяется функциональная модель акторского, агентского, функционального, объектного и процедурного языков программирования. Функциональная модель позволяет сравнивать широкий спектр ранее разрозненных стилей языка программирования и разработана для расширения.

Пять языковых сопоставлений, используемых для проверки модели, должны быть полезны для практиков при оценке пригодности для конкретного проекта разработки или архитектуры программного обеспечения. Тем не менее, функции являются лишь одним из многих факторов, которые необходимо учитывать при принятии таких решений. Ключевым критерием является наличие специалистов с достаточным опытом работы на выбранном языке. Другие факторы потенциальной важности включают совместимость платформы, поддержку средств разработки, документацию, учебные материалы, сообщество пользователей и нефункциональные свойства. Некоторые важные концепции языка программирования не были явно включены в текущую функциональную модель.

Область видимости. Ван Рой и Хариди определяют область видимости как «часть текста программы, в которой элемент [a] виден». Область действия обычно выражается в терминах конкретных языковых конструкций, что затрудняет сравнение правил области действия между языками.

Исключения. Ошибка или исключение во время выполнения определяется как непредвиденное условие, которое не может быть обработано локально. Правила определения, распространения и восстановления исключений обязательно зависят от текущего контекста; как и область действия, отказ - это сквозная проблема, которая нелегко поддается модели, основанной на функциях.

Безопасность. Безопасность, определяемая как «защита как от вредоносных вычислений, так и от невинных (но ошибочных) вычислений», является глобальным системным свойством. В то время как определенные языки

имеют известные недостатки безопасности, современные механизмы безопасности обычно реализуются компилятором, интерпретатором, виртуальной машиной или операционной системой.

Именование. Такие языки, как «Haskell», предлагают сложные возможности именования для сопоставления с образцом, однако их сложно сопоставить с моделью объектов, поскольку они охватывают несколько групп объектов, таких как системы типов и декларативные выражения. Большинство языков позволяют назначать одновременные действия двум или более отдельным узлам.

Наконец, модель объектов не включает никаких оценочных суждений о наличии или отсутствии языковых функций. Мы утверждаем, что ценность данной функции по своей природе зависит от приложения; «Простое наличие функций не является хорошим показателем ценности языка». Полнофункциональный язык позволит кратко выразить широкий спектр программ, но за счет более дорогой реализации и более сложной кривой обучения.

Основная исследовательская ценность представленной функциональной модели заключается в будущей работе, которую она дает. Часть этой работы изложена следующим образом.

Сопоставление других языков программирования с моделью объектов позволило бы ее доработать и утвердить в дальнейшем.

Модель предназначена для использования в качестве основы для структурированных сравнений, в том числе эмпирических сравнений, между языками программирования в любом из типов: актор, агент, функционал, объект и стиль. Эта работа потребует разработки объективных критериев, чтобы определить, присутствует ли каждая особенность или отсутствует.

В более долгосрочной перспективе, учитывая достаточное понимание областей приложения, в которых обычно используются субъектный, агентный, функциональный, объектный и процедурный языки программирования, значения функций в модели могут быть определены как функции характеристик домена. Аппроксимации этих функций можно было бы получить эмпирически, изучая использование функций в существующих приложениях или экспериментируя с пустяковыми проблемами. Знание того, какие функции желательны, помогло бы тогда разработчику или специалисту создать или выбрать язык программирования, который соответствует конкретной области приложения.

Библиографический список

1. Jordan H. A feature model of actor, agent, functional, object, and procedural programming languages / H. Jordan, G. Botterweck, J. Noll, A. Butterfield, R. Collier. – Science of Computer Programming, 2015. – Volume 98, Part 2 – Pp. 120-139.