Использование визуального языка на основе REAL.NET для программирования потоков данных

Кидянкин М. В., студент математико-механического факультета СПбГУ,
kidyankin@icloud.com

Аннотация

Визуальные технологии могут найти свое применение в процессе разработки различных систем. В данной статье приводится описание использования визуальных языков для программирования потоков данных. Описаны компонентов решения, таких как визуальный язык и генератор кода, а также инструментов их реализации. Особое внимание уделено применимости предлагаемых подходов и положительным эффектам от использования подобных технологий.

Введение

Визуальное программирование находит большое количество применений в различных областях, например, при обучении программированию или при программировании роботов [1]. Однако визуальное моделирование может выступать в роли полезного инструмента и для программиста. Например, с помощью визуального языка программирования можно легко описать структуру работы с данными в модели потоков (flows), где вычисления представляются в виде блоков и связей между ними.

Одним из примеров библиотеки для работы с потоками данных может выступать TPL Dataflow [2] платформы Microsoft .NET. Помимо обеспечения программирования в терминах блоков и связей, она позволяет автоматически и безопасно распараллеливать обработку данных. Однако библиотека содержит некоторые ограничения, что может увеличить количество блоков в модели и усложнить её. Кроме этого, описание большого количества блоков и их связей линейно в коде приводит к появлению ошибок, не всегда легко обнаруживаемых. В статье приводится описание того, как применение специального визуального языка и генератора может облегчить процесс описания систем управления потоками данных.

Текущие решения

Безусловно, для описания потоков данных уже существуют визуальные языки, например, широко используемая в программировании аппаратных устройств система LabVIEW и её визуальный яызк G [3], однако распространенные решения для работы с TPL Dataflow и дальнейшей генерации в исходный код на C# не были найдены.

Полезным инструментом может послужить TPL DataFlow Debugger Visualizer [4], позволяющий по готовому решению посмотреть его визуальное представление. В данной статье рассматривается решение, позволяющее производить обратную операцию — по визуальной модели создавать готовое решение в виде исходного кода.

Обзор используемых технологий

TPL.Dataflow

Как было отмечено ранее, библиотека TPL.Dataflow [2] позволяет описывать вычисления в модели блоков, осуществляющих операцию над данными, и связей, по которым данные передаются между разными блоками. Блоки делятся на три типа: *блоки источника*, из которых можно считать данные, *целевые блоки*, выступающие в роли конечного получателя данных, *блоки передачи*, получающие некоторые данные из других блоков, выполняющие на ними действия и передающие обработанные данные следующим блокам. Блоки линейно соединяются в конвейер, либо в более сложные сети с использованием дополнительных блоков-оберток, позволяющих передавать данные нескольким блоками и группировать получаемые из нескольких источников данные для передачи в один.

REAL.NET

REAL.NET [5] представляет собой среду для визуального моделирования, разрабатываемую на кафедре Системного программирования СПбГУ. В отличие от многих подобных решений, она реализует принцип ”глубокого метамоделирования” – относительно нового подхода к описанию визуальных языков, позволяющего проводить описание языков семантически более чисто, а также избежать возможным проблем, возникающих при использовании распространенной двухуровневой схемы. На данный момент реализовано несколько редакторов разной функциональности под разные операцион-

ные системы. Проект активно развивается, добавляется новая функциональность, расширяются сферы возможного применения.

Razor

ASP.NET Razor представляет собой инструмент, применяющийся для генерации представления из шаблона и модели с данными. Обычно он применяется для генерации веб-страниц, но также может быть использован, например, при генерации файлов исходного кода. Ввиду популярности Razor для платформы .NET было принято решение использовать его, а точнее его легковесную реализацию RazorLight [6].

Описание решения

Визуальный язык

Поскольку вычисления представляются в виде блоков и связей, структура визуального языка получается достаточно простой. Заданы узлы (блоки), которые представляют собой некоторое действие или функцию. В качестве параметров этих узлов выступают тип выходного значения и имя блока. Связи между узлами указывают, в какой блок полученные результаты должны быть отправлены как входные значения. Связи обязательно устанавливаются между двумя блоками, при этом из одного блока могут выходить несколько связей (если необходимо передать данные в несколько других узлов) и могут входить несколько связей (если для выполнения действия в узле требуются данные из нескольких источников).

Генерация кода

Ограничения библиотеки TPL Dataflow являются более строгими, чем у описанного выше визуального языка. Например, блоки действий в TPL Dataflow могут иметь только один вход и выход. Если программист хочет реализовать множественные связи, то их необходимо дополнительно обернуть в другие специальные блоки. Таким образом, реальное количество блоков и связей может быть больше, чем в модели, описанной в визуальном языке. Чтобы упростить описание таких схем, язык допускает множественные связи, которые затем автоматически обрабатываются генератором в корректные для библиотеки.

Кроме этого, генератор автоматически определяет тип входных данных для каждого блока и на основании этих данных выбирает тип блока, описанного в TPL Dataflow (например, блоки-действия, не возвращающие значения, и блоки-функции имеют разный тип, однако программист его не специфицирует во время описание модели на визуальном языке). Таким образом, благодаря подобным оптимизациям на основе анализа схем можно добиться ускорения и упрощения разработки потоков данных.

Диаграмма компонентов решения приведена на Рис. 1.

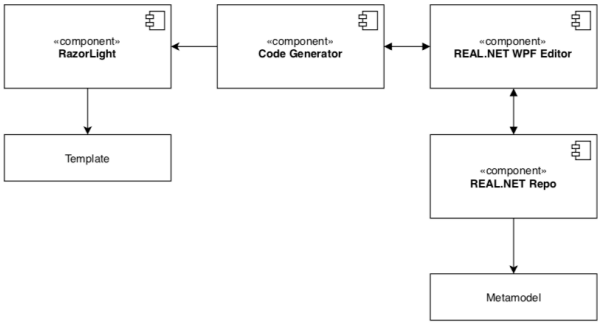

Рис. 1: Диаграмма компонентов решения

Применимость

В качестве примера работы описываемого решения приведем следующую задачу. Пусть даны две точки на карте, по которым нам нужно найти самый быстрый путь между ними, используя перемещения на поезде или на самолете. По двум заданным точкам для каждой их них мы найдем ближайший к этой точке город (эти два действия можно выполнять параллельно), а затем найдем два возможных маршрута между городами (это тоже можно делать параллельно). Два подобранных результата выводятся пользователю. Описание этой задачи в терминах потоков данных на приведенном визуальном языке проиллюстрировано на Рис. 2. В результате работы предлагаемого инструмента по данной схеме была сгенерирована 41 строка исходного кода, содержащая 4 дополнительных блока, не описываемых в модели на визуальном языке, 12 связей. Аналогичная задача была реализована без использования генератора и визуального языка, было проведено сравнение результатов их работы на одинаковых тестах, в результате которого установлены одинаковые результаты у обоих решений.

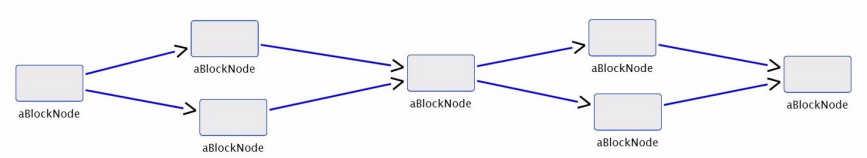

Рис. 2: Фрагмент схемы на визуальном языке

Кроме этого, в рамках подготовки работы было проведено анкетирование, где участникам предлагалось оценить различные подходы к описанию таких систем. Так, большинство анкетируемых (более 90%) отметили, что начнут проектирование, нарисовав схему на бумаге, вместо описания ее в коде. Кроме этого, все участники отметили, что для ознакомления с уже спроектированной системой управления потоками данных они предпочтут визуальную схему вместо исходного кода. По результатам опроса участники хотели бы протестировать данный генератор при необходимости описать потоки данных, что подтверждает возможности применения визуального программирования в данной области.

Заключение

Приведенное выше решение, помимо прочего, создано продемонстрировать возможности применения визуального программирования в разработке. Преимущества, предоставляемые данным инструментом, заключаются в возможности упрощенного использования библиотеки программирования потоков данных, что уменьшает время, затрачиваемое на разработку, уменьшает вероятность ошибки за счет анализа схемы, а также повышает читаемость подобных схем и уменьшает время, необходимое на ознакомление с уже спроектированной системой. Ознакомиться с инструментом можно в GitHub-репозитории проекта [7].

Литература

[1] Dmitry Mordvinov, Yurii Litvinov, Timofey Bryksin. TRIK Studio: Technical Introduction // Proceedings of the FRUCT'20. — 2017. — pp. 296-308.

[2] Официальная документация библиотеки Dataflow (Task Parallel Library) URL:https://docs.microsoft.com/dotnet/standard/parallel-programming/dataflow-task-parallel-library (дата обращения: 04.04.2019)

[3] Travis J., Kring J. LabVIEW for everyone: graphical programming made easy and fun. – Prentice-Hall, 2007.

[4] Offir Shvartz. TPL DataFlow Debugger Visualizer

URL:https://marketplace.visualstudio.com/items?itemName=OffirShvartz.TPLDataFlowDebuggerVisualizer (дата обращения: 04.04.2019)

[5] Литвинов, Ю. В., Кузьмина, Е. В., Небогатиков, И. Ю., Алымова, Д. А. (2017). Среда предметно-ориентированного визуального моделирования REAL.NET. // Материалы 7-й всероссийской научной конференции по проблемам информатики СПИСОК-2017. — 2017. — стр. 80-89.

[6] RazorLight – библиотека для использования Razor

URL:https://github.com/toddams/RazorLight (дата обращения: 04.04.2019)

[7] Визуальный язык для программирования потоков данных TPL Dataflow

URL:https://github.com/mikekidya/REAL.NET (дата обращения: 04.04.2019)