Тензор кручения – $(p_{12}-q_{11}-1,0,0),(p_{13}-r_{11},0,0),(0,q_{23}-r_{22},q_{11}-p_{12}+1)$. Прямыми вычислениями получаем, что пара не допускает эквиаффинных связностей. Таким образом, в работе определено, при каких условиях пара не допускает эквиаффинных связностей.

ЛИТЕРАТУРА

1. Nomizu, K. Affine differential geometry / K. Nomizu, T. Sasaki. – Cambridge Univ. Press, 1994. – 263 p.

2. Онищик, А. Л. Топология транзитивных групп Ли преобразований / А. Л. Онищик. – М. : Физ.-мат. лит., 1995. – 384 с.

3. Кобаяси, Ш. Основы дифференциальной геометрии : в 2 т. / Ш. Кобаяси, К. Номидзу. – М. : Наука, 1981. – 2 т.

4. Можей, Н. П. Трехмерные редуктивные пространства разрешимых групп Ли / Н. П. Можей // Известия Гомельского государственного университета имени Франциска Скорины. – 2016. – № 6 (99). – С. 74–81.

Г. Л. МУРАВЬЕВ, С. В. МУХОВ, В. И. ХВЕЩУК

УО БрГТУ (г. Брест, Беларусь)

О ПРОТОТИПИРОВАНИИ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ДИАГРАММ UML

Объект рассмотрения – результаты объектно-ориентированного анализа предметной области проектов программ (объектная модель в терминах диаграмм UML [1; 2]) в части их использования для прототипирования [3] оконных приложений, что производится путем специализации типовых каркасов приложений, поддерживаемых системой программирования (на примере Microsoft Visual Studio), с учетом предъявляемых функциональных требований.

При проектировании программ основываются на требованиях, которые предъявляются к свойствам конечного продукта. В первую очередь, это функциональные требования, задаваемые множеством реализуемых задач-функций, отображающих входные данные в выходные и определяющих закон функционирования программы. Также это требования к характеристикам исполнения, а для оконных программ – к дизайну и организации интерфейсов и др.

Рынок современных программных продуктов отличается значительной долей «человеко-ориентированных» программ, характеризуемых развитым графическим интерфейсом (ГИП). Как правило, это класс приложений, управляемых событиями (event-driven applications). Для них известны установившиеся тенденции, стратегии проектирования в таких значительных сферах применения, как экономика, социальные учреждения, индивидуальные потребители и т. п. [1–3].

Это акцент на продуктивности разработки, достигаемой зачастую в ущерб характеристикам эффективности использования программ (производительности,

ресурсо-потреблению и т. п.), что не мешает после получения функционально-полного продукта улучшать эти характеристики рядом мероприятий, последующей оптимизацией кода, не затрагивающей достигнутую функциональность.

Указанное делает результативным применение таких элементов технологий активной разработки [4] программ, как:

– опережающее моделирование продукта по результатам анализа предметной области, автоматизируемых задач;

– раннее вовлечение в процессы проектирования заказчиков, использование их знаний для анализа проблемы, функций системы, тестирования решений.

Это предполагает выполнение прототипирования – создание действующей модели, прототипа проекта, в том числе путем нисходящего проектирования и пошаговой детализации [5]. Тем самым обеспечивается получение семейства прототипов разной степени готовности, состоящих из моделей компонентов (от компонентов-заглушек до готовых компонентов) проекта (подпрограмм, методов, обработчиков), которые с разной степенью детальности отображают исполняемые функции.

Рассматриваемый класс приложений представляют собой наборы графических ресурсов интерфейса (окна, элементы окон), поддерживаемого функциями-обработчиками, связанными с соответствующими окнами и обслуживающими события, происходящие в системе. Они же совместно с методами классов предметной области обеспечивают нужную функциональность (бизнес-логику) продукта. Такие программы:

– представляют собой динамическую систему, отличаются “автоматным” стилем поведения, когда функционирование приложения представляет собой процессы изменения состояний под влиянием происходящих событий;

– в качестве их математического описания могут использоваться детерминированные автоматы как модели с дискретным временем и конечным множеством состояний, ассоциируемых здесь – на системном уровне – с тем набором окон интерфейса, которые могут получать активность, в том числе брать на себя «фокус» ввода и поддерживать действия пользователя;

– активизация окна означает переход в новое состояние, связанное с ожиданием последующих сообщений. То есть сообщения, в том числе о действиях пользователя, являются событиями – причинами изменения состояния;

– процесс изменения состояния (включая возврат в исходное состояние) сопровождается запуском функции-обработчика, который обеспечивает заданный сценарий действий, поток событий.

В качестве средства внутреннего представления исходных данных для выполнения прототипирования рассматриваются модели языка UML, включая модели состояний, прецедентов и др. Они обеспечивают:

– формализацию результатов анализа проекта;

– создание структурированной информационной базы (иерархии классов), обеспечивают обработку описаний;

– автоматизацию получения прототипов.

Дополнительные возможности в части детализации и построения структурированных описаний потоков событий прецедентов, обработчиков, методов может обеспечить применение диаграмм классов, объектов, видов деятельности, учитывающих

привязку действий, операций к соответствующим классам, включая классы поддержки манипуляций с окнами, сообщениями и т. п.

Следует также учесть рост составляющей рутинных операций, функций в разработке программ (например, в части работы с интерфейсами, диспетчеризации сообщений и т. п.), которые образуют наращиваемый далее пользователем каркас-приложения. Поэтому современные среды разработки программ поддерживают каркасное программирование – автоматическую генерацию типовых авто каркасов с учетом языка программирования, платформы, библиотек и дополнительных пользовательских настроек.

Это создает предпосылки для автоматизации построения прототипов программ [6]. Здесь прототип как модель программы (проекта) представляет собой исполнимый, специализированный с учетом требований к продукту, его функциональности каркас, построенный на базе выбранного типового авто каркаса.

В работе рассмотрены проблемы и подходы к генерации прототипов проектов, результативность применения диаграмм UML. Показана возможность расширения типовых каркасов путем их дооснащения:

– меню;

– окнами, иерархиями окон, элементами управления в составе интерактивных окон;

– методами и прототипами методов-обработчиков сообщений и методов поддержки бизнес-логики приложения;

– шаблонами и иерархиями пользовательских классов.

Результаты могут использоваться для автоматизации процессов генерации прототипов проектов в целях повышения результативности этапов анализа и проектирования программ. Способствуют повышению эффективности обучения проектированию приложений, анализу предметных областей.

ЛИТЕРАТУРА

1. Липаев, В. В. Программная инженерия. Методологические основы : учеб. / В. В. Липаев. – М. : ТЕИС, 2006. – 608 с.

2. Мяцяшек, Л. А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML : пер. с англ. / Л. А. Мяцяшек. – М. : Издательский дом «Вильямс», 2022. – 432 с.

3. Орлов, С. А. Программная инженерия / С. А. Орлов. – СПб. : Питер, 2016. – 640 с.

4. Активное программирование [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/. – Дата доступа: 10.10.2021.

5. Эванс, Э. Предметно-ориентированное проектирование: структуризация сложных программных систем / Э. Эванс. – М. : ООО «И. Д. Вильямс», 2016. – 448 с.

6. Муравьев, Г. Л. Об исполнимости спецификаций проектов / Г. Л. Муравьев, С. В. Мухов, В. И. Хвещук // Инновационные технологии обучения физико-математическим и профессионально-техническим дисциплинам : материалы 8-й междунар. научно-практ. конф., Мозырь, 22–25 марта 2016. – С. 153–156.