УДК 004.025.2:510.652

ОБЕСПЕЧЕНИЕ НАДЕЖНОСТИ ПРОГРАММНЫХ СРЕДСТВ ПРИ ИХ МОДИФИКАЦИЯХ

Н.В. УСПЕНСКАЯ[1], В.В. БАХТИЗИН[2]

[1]Белорусский государственный университет информатики и радиоэлектроники
ул. П. Бровки, 6, г. Минск, 220013, Республика Беларусь
nvuspenskaya@gmail.com

[2]Белорусский государственный университет информатики и радиоэлектроники
ул. П. Бровки, 6, г. Минск, 220013, Республика Беларусь
bww@bsuir.by

Использование программных средств для моделирования сложных систем в последнее время пользуется все большей популярностью. В этой связи большую важность представляют методы обеспечения надежности программных средств для моделирования сложных систем. Одним из средств для вышеуказанной цели является проектирование по контракту.

Ключевые слова: проектирование по контракту, предусловия, модифицированные программные средства.

Для обеспечения надежности программных средств (ПС) для программного моделирования сложных систем предлагается использовать проектирование по контракту (Design by Contract). Данный способ устанавливает отношение между модулями программного средства, однозначно устанавливающий права и обязанности сторон. Такой подход обеспечивает точное определение требований и ответственности для каждого модуля, что повышает тестируемость и диагностируемость программных средств, что в свою очередь обеспечивает их надежность [1].

Основная идея контрактного программирования — это модель взаимодействия элементов ПС. Взаимодействие между модулями ПС происходит по схеме “клиент-поставщик” в соответствии с определённым контрактом. Контракт некоторого объекта включает в себя обязательства клиентского модуля (предусловия), обязательства, присутствующие после выполнения метода (постусловия), обязательства по выполнению конкретных свойств — инвариантов, которые должны выполняться при получении поставщиком сообщения, а также при выходе из метода [2].

Любая открытая функция класса – это не просто абстрактный фрагмент кода, предназначенный для выполнения неопределенной задачи. Каждая функция обладает строгими семантическими свойствами, которые отражают, что делает эта функция, независимо от того, как она это делает. Для четкого выражения задачи, выполняемой конкретной функцией, с ней связывают два утверждения – предусловие и постусловие. Предусловие определяет условия, которые должны выполняться всякий раз перед вызовом функции, а постусловие определяет свойства, гарантируемые после завершения ее выполнения. Предусловие и постусловие определяют контракт функции со всеми ее клиентами [2].

Нарушением контракта является отклонение конкретной реализации от заданной спецификации. При включенном мониторинге нарушение утверждения в период выполнения приводит к генерации исключения.

Выбор уровня мониторинга зависит от разработчика. Предлагается использовать максимальный уровень мониторинга как при отладке ПС, так и в модулях, критичных к высокой надежности выполнения операций. Для модулей, в которых критично время

выполнения операций, предлагается рассчитывать уровень мониторинга в зависимости от производительности приложения.

В случае значительного падения показателей производительности ПС, предлагается не отключать мониторинг полностью и обязательно проверять предусловия. Опыт показывает, что это значительно менее критично к ресурсам нежели проверка постусловий и обеспечивает достаточно высокий уровень надежности по сравнению с полным отсутствием мониторинга.

Как показывает практика проектирования ПС, контрактное программирование повышает уровень повторного использования кода и позволяет однозначно определить причину проблемы в повторно используемом коде.

Существуют слабые и сильные предусловия. Понятия «сильный» и «слабый» пришли из логики. Говорят, что условие *P1* сильнее, чем *P2*, а *P2* слабее, чем *P1*, если выполнение условия *P1* влечет за собой выполнение условия *P2*, но они не эквивалентны, поскольку при выполнении условия *P1* выполняется и условие *P2*,при этом эти условия не эквивалентны [2].

Любое условие, заведомо равное True, является слабейшим, а условие, заведомо равное False, является сильнейшим из всех возможных описаний. Необходимо избегать данных крайних значений. В случае слабейшего условия, контракт будет выполнен в любом случае, что в корне неприемлемо в соответствии с теорией контрактного проектирования. В случае сильнейшего предусловия контракт будет невыполнен вечно, что навредит функциональной эксплуатации ПС.

В описании метода проектирования по контракту не описаны подходы для определения степени «силы» для предусловия. Предлагается реализовывать предусловия выполнения контракта максимально сильными, но не доходяшими до крайних значений. Т.е. условия выполнения контракта должны предусматривать проверку корректности всех внешних данных, используемых при выполнении метода. Значения для проверки должны быть минимально удовлетворяющие контракту. Это позволит обеспечить высокую надежность и при этом не повредит работоспособности ПС.

Определить степень того, насколько предусловие сильно и подходит для конкретного случая проверки контракта предлагается исходя из того, что успешность выполнение метода не должно зависеть от состояния данных после проверки предусловия. При выполнении данного условия ожидается, что ответственность за сбой в работе лежит на модуле, который выполнил последнюю операцию, что вносит ясность при расследовании причин неправильной работы ПС.

Для обеспечения целостности данных рекомендуется выполнять проверку контракта методами, реализованными в классе-источнике. Опыт показывает, что стратегия, при которой каждый модуль ПС несет ответственность только за свои действия, приносит результаты при эксплуатации, сопровождении и доработке модифицированных ПС.

Как показывает практика проектирования ПС, контрактное программирование повышает уровень повторного использования кода и позволяет однозначно определить причину проблемы в повторно используемом коде.

Список литературы

1. *Мейер, Б.* Объектно-ориентированное конструирование программных систем. М.: Русская редакция, 2005.

2. *Meyer, B.* Touch of Class. Leaning to Program Well with Objects and Contracts. London: 2009.