ЭВОЛЮЦИОННОЕ РАСШИРЕНИЕ ПРОГРАММ ПРИ РАЗЛИЧНЫХ ПАРАДИГМАХ ПРОГРАММИРОВАНИЯ

Легалов А. И., Легалов И. А., Солоха А. Ф.
(г. Красноярск, Сибирский федеральный университет)

1. Введение

Эволюционная разработка характеризуется добавлением новых программных объектов в уже написанный код. Эти добавления могут быть комплексными и порождать разнообразные комбинации. Можно выделить ряд типичных ситуаций расширения кода, часто встречающихся на практике [1]:

- расширение обобщений специализациями и, как следствие, расширение обрабатывающих их обобщающих процедур;
- добавление новых процедур, обеспечивающих дополнительную функциональность;
- добавление новых полей данных в существующие типы и изменение в соответствии с этим процедур, осуществляющих обработку измененных программных объектов;
- добавление новых процедур, предназначенных для обработки только одной из специализаций некоторого обобщения;
- создание нового обобщения на основе существующих специализаций;
- добавление в программу мультиметодов, осуществляющих обработку двух или более обобщенных параметров;
- изменение мультиметодов при добавлении ноавых специализаций в обобщения, используемые в качестве аргументов мультиметодов.

Эти варианты могут встречаться как раздельно, так и совместно. В последнем случае порождаются разнообразные комбинации, реализация которых требует нетривиальных алгоритмических решений. Сложность и гибкость этих решений зависит от используемой парадигмы программирования, которая в ряде случаев позволяет реализовать требуемую ситуацию с применением прямого использования предоставляемых ею инструментальных средств.

Однако существуют ситуации, когда требуемое решение достигается за счет использования дополнительных программных решений, реализующих требуемый конструктив за счет использования множества взаимодействующих между собой программных объектов. Наиболее удачные из таких решений часто оформляются в виде паттернов [2]. В работе рассматриваются особенности прямой реализации описанных выше простых ситуаций для трех парадигм программирования: процедурной, объектно-ориентированной и процедурно-параметрической. Анализируется возможность расширения программы без изменения уже написанного кода. Основной задачей является исследование возможностей процедурно-параметрической (ПП) парадигмы программирования [3-5] и его сравнения с возможностями объектно-ориентированного (ОО) и процедурного подходов.

2. Добавление в обобщение новых специализаций

Процедурный подход. В традиционных процедурных языках для построения обобщений обычно используются объединения (в языке C [6]) или вариантные записи (языки Pascal [7], Modula-2 [8]), что не обеспечивает эволюционного расширения и требует модификации уже написанного кода. Безболезненного расширения данных в этих языках можно достичь использованием указателя на произвольный тип, что снижает безопасность

программного кода [1] из-за появляющейся возможности некорректного использования такого указателя.

Использование при процедурном подходе расширений типов данных обеспечивает безболезненное добавление новых специализаций в таких языках программирования как Оберон [8-9], Оберон-2 [9], Component Pascal [10], Ada [11]. Аналогичным образом проблема расширения решается и в языке C++ [12] применением наследования к структурам.

Добавление новых специализаций в существующие обобщения ведет к необходимости добавления обработчиков обработчиков этих специализаций, которые обычно реализуются как ветви условного оператора или эквивалентного ему оператора выбора (переключения), вставляемые в имеющиеся процедуры, обрабатывающие это обобщение, что не способствует эволюционному расширению написанного кода.

Объектно-ориентированный подход. Для добавления специализации в языках программирования Оберон-2, Component Pascal и Ada применяется расширение типа. Для добавления новых обработчиков специализаций используются процедуры, связанные с типом. В C++, C# [13], Java [14] и других объектно, ориентированных языках в качестве специализации создается производный класс, а обработчики специализации реализуются посредством виртуальных методов, расширяющих обобщающий виртуальный метод базового класса. Таким образом, ООП поддерживает безболезненное расширение в данной ситуации.

Процедурно-параметрический подход. Обеспечивает добавление новых специализаций через подключение специализации к обобщению за счет существующей в языке внешней операции добавления [3-5]. Необходимые обработчики специализаций также легко расширяют уже существующие обобщающие процедуры за счет дописывания новых реализаций.

3. Добавление процедур с дополнительной функциональностью

Процедурный и процедурно-параметрический подходы. Использование внешних процедур обеспечивает простое решение данной задачи. Не возникает никаких проблем с их безболезненным добавлением в новых единицах компиляции. Подобный подход успешно применим практически во всех языках процедурного программирования.

Объектно-ориентированный подход. Метод, реализующий дополнительную функциональность приходится вставлять внутрь класса, что ведет к изменению интерфейса последнего. Сама реализация метода в ряде языков (C++) может быть описана в новой единице компиляции, что уменьшает объем изменений вносимых в программу. Однако большинство современных ОО языков (C#, Java), ориентированных на автоматическую генерацию интерфейсов модулей, используют запись реализации методов внутри класса, что увеличивает объем изменений уже написанного кода. Однако, если изменяемый класс является базовым, даже при изменении только интерфейса произойдет перекомпиляция всех модулей, которые зависят как от этого класса, так и его производных классов. Поэтому, при разработке объектно-ориентированных программ рекомендуется большее внимание уделять методам, образующим интерфейсы классов, чтобы впоследствии избежать их изменения [15]. Таким образом, ОО парадигма напрямую не поддерживает эволюционного добавления новых методов, расширяющих функциональность класса.

4. Добавление новых полей в существующие типы

Прямое добавление новых полей в запись или класс ведет к его изменению независимо от избранного стиля программирования. Поэтому, при необходимости модификации типа с сохранением возможности безболезненно расширять программу, используются обходные пути, опирающиеся на косвенное связывание. В этом случае

осуществляется связывание через указатель на базовый тип, к которому подключается объект производного типа, содержащий дополнительные поля. При обработке такого объекта необходимо использовать явное приведение типа или дополнительные функции. Если подобные действия приходится применять в ранее написанных модулях, то эволюционное расширение программы невозможно. Однако часто программа, использующая модифицированный тип, может быть расширена таким образом, что его обработка осуществляется только в добавленных единицах компиляции. Поэтому данный подход может использоваться для всех трех парадигм программирования с применением присущих им технических приемов.

Процедурный подход. Для добавления полей косвенным связыванием в языке программирования Оберон удобно пользоваться расширением типов. В этом случае новый тип расширяет предшественника и через указатель на базовый тип может использоваться по целевому назначению. Реализованный в языке механизм проверки типа во время выполнения позволяет легко перейти от базового типа к производному. Поэтому можно считать, что поздние процедурные языки поддерживают добавление новых полей используя полиморфизм типа.

Вместе с тем, следует отметить, что традиционные процедурные языки (C, Pascal) не обеспечивают реализацию подобного приема из-за отсутствия дополнительной инструментальной поддержки проверки типа. Поэтому их использование в таком режиме невозможно без написания дополнительного кода.

Объектно-ориентированный подход. Добавление новых полей, как и при расширении типов, осуществляется в производном классе. При использовании языков, поддерживающих динамическую идентификацию типа во время выполнения (C#, Java, Object Pascal [7]), использование классов аналогично применению расширений типов, описанному для процедурных языков. В этом случае, наряду с данными можно включать и новые методы, осуществляющие их обработку. Вместе с тем, следует отметить, что использование явной проверки типа во время выполнения в большей степени является процедурным, чем объектно-ориентированным приемом, так как объектно-ориентированный полиморфизм базируется на сочетании наследования и виртуализации.

При отсутствии динамической проверки типа или отключения этого режима (например, в C++) использование косвенного добавления полей в класс становится невозможным. Для его реализации необходимы дополнительные алгоритмические приемы, например, включение в базовый класс виртуального метода, переопределяемого в производных классах, за счет чего обеспечивается требуемая идентификация. Таким образом, использование объектно-ориентированного полиморфизма в данной ситуации невозможно без дополнительной алгоритмической поддержки.

Процедурно-параметрический подход. Использование косвенного добавления полей допускается при использовании обобщенных записей. В этом случае расширение заключается в добавлении специализации. Доступ к специализации поддерживается механизмом проверки признака специализации, по которому можно установить ее тип.

5. Добавление новых процедур для обработки конкретных специализаций внутри обобщений

Данная ситуация возникает, когда необходимо использовать только одну из специализаций, например, при обработке элементов контейнера, ссылающегося на обобщенный тип, через который нужно обеспечить доступ только к одному из специализированных типов.

Процедурный подход. В случае расширений типа используются процедуры, в которых осуществляется явная проверка производного типа по указателю на базовый тип [1].

Метод легко добавляется в новом модуле. Не вызывает проблем использование этого же приема в процедурных языках, не имеющих поддержки полиморфизма типа (например, в С). В этом случае обобщение обычно строится таким образом, что имеет признак, который и используется при идентификации специализации.

Объектно-ориентированный подход. Применение объектно-ориентированных методов опирается на использование виртуальных функций, осуществляющих обработку специализированного объекта требуемого типа [1]. Однако такая функция, для использования виртуализации, должна добавляться в базовый класс, что не способствует эволюционному расширению программы. Помимо этого, базовый класс перегружается дополнительной информацией о производном классе, что не способствует инкапсуляции. Поэтому для обработки специализаций при объектно-ориентированном подходе чаще всего применяется процедурный прием, основанный на явной проверке типа.

Процедурно-параметрический подход. Простейшим вариантом является использование процедурного подхода, примененного к обобщенному типу [3]. Отличие заключается в проверке признака специализации вместо проверки производного типа подключенного объекта. Поэтому при процедурно-параметрическом подходе не возникает проблем в решении данной задачи.

Однако, помимо этого, возможно использование процедурно-параметрического полиморфизма, при котором отсутствует необходимость явной манипуляции типом объекта. В этом случае в начале пишется обобщающая процедура с пустым телом (как и при объектно-ориентированном подходе). Для выполнения требуемых манипуляций с заданной специализацией пишется ее обработчик, который и вызывается при вызове обобщающей процедурой. В отличие от ОО подхода все добавления осуществляются в новых модулях, что обеспечивает эволюционное расширение программы. Таким образом, процедурно-параметрическая парадигма предоставляет разнообразные способы для разрешения данной ситуации.

6. Создание обобщения из существующих специализаций

Процедурный подход. Зачастую в программе требуется сформировать новое обобщение при уже существующих специализациях. Процедурный подход в этом случае обеспечивает прямое решение на основе объединения (язык С) или вариантной записи (языки Pascal, Modula-2). Данный тип может формироваться в модулях, добавляемых в программу, что не вызывает проблем с эволюционным расширением. Вместе с тем, следует отметить, что языки, использующие для построения обобщений только расширение типа (Oberon, Oberon-2), не обеспечивают прямого решения этой проблемы. Необходимы дополнительные построения, чтобы достичь искомого решения.

Объектно-ориентированный подход. Использование наследования обладает теми же недостатками, что и расширение типа. Поэтому построение обобщения обычно ведется на основе создание нового базового класса и порождения от него производных классов, включающих в себя необходимые существующие специализации. Решение достаточно простое, но не является прямым.

Процедурно-параметрический подход. Обобщения и обобщенные записи при своем создании допускают непосредственное включение в них специализаций, что обеспечивает их прямое построение. Помимо этого, расширение за счет уже существующих специализаций возможно и при уже существующем обобщении. Это обеспечивает высокую гибкость и эволюционный рост для рассматриваемой ситуации.

7. Добавление мультиметодов

Процедурный подход. Так как мультиметод является обычной процедурой, его добавление не вызывает никаких проблем. Следовательно, процедурный подход поддерживает эволюционное добавление мультиметодов.

Объектно-ориентированный подход. Реализация мультиметода в объектно-ориентированном подходе опирается на множественную диспетчеризацию [16], при которой между классами, играющими роль обобщенных аргументов мультиметода, возникают тесные взаимодействия, задаваемые через методы. Поэтому добавление мультиметода ведет к тому, что изменяется описание класса, что не способствует эволюционному расширению. В связи с тем, что использование диспетчеризации ведет к большим изменениям во взаимодействующих классах, обычно при реализации мультиметодов применяются подходы, базирующиеся на явной проверке типов или смешанном варианте, когда тип первого аргумента выявляется полиморфно, а тип второго определяется явной проверкой [17, 18]. Однако подобные решения все равно не обеспечивают поддержку эволюционного расширения, так как приходится изменять содержимое класса.

Процедурно-параметрический подход. Для гибкой реализации мультиметодов используются обобщающие параметрические процедуры [4, 5]. Являясь по сути внешними процедурами, они обеспечивают безболезненное добавление мультиметодов в программу.

8. Изменение мультиметодов при добавлении специализаций

Процедурный подход. Выбор обработчика комбинации специализаций мультиметода осуществляется на основе явной проверки типов обобщенных аргументов внутри условных операторов или операторов выбора. Поэтому добавление новой специализации к любому обобщенному аргументу мультиметода изменяет один или несколько условных операторов, не обеспечивая эволюционного расширения программы в данной ситуации [1].

Объектно-ориентированный подход. Добавление новой специализации связано с созданием нового производного класса [1]. Включение этого класса в общую группу, содержащую мультиметод, ведет, для обеспечения диспетчеризации, к добавлению дополнительных методов во все классы группы. Поэтому прямое добавление новых специализаций в мультиметод не поддерживается. Стоит отметить существование алгоритмических решений, обеспечивающих безболезненное расширение мультиметодов в частных случаях на основе объектно-ориентированного полиморфизма [17-20] в основном за счет сочетания процедурного и объектно-ориентированного подходов.

Процедурно-параметрический подход. Данная парадигма изначально разрабатывалась для поддержки безболезненного расширения мультиметодов при добавлении новых специализаций [3]. Это обеспечивается за счет описания всех комбинаций аргументов в обработчиках специализаций, размещаемых независимо от обобщающей процедуры. Поэтому каждая новая специализация ведет лишь к написанию соответствующих обработчиков в новых единицах компиляции [4, 5]. Метод основан на достаточно простом механизме построения параметрических отношений, одна из возможных реализаций которого приведена в [18].

9. Общая характеристика подходов

В таблице собраны сведения, показывающие возможности каждой из рассмотренных парадигм по прямой поддержке эволюционного расширения. Можно увидеть, что ОО подход обеспечивает прямое эволюционное расширение программы в наименьшем числе из

рассмотренных ситуаций. Однако его текущая популярность вполне объяснима. Во-первых, ОО парадигма обеспечивает добавление новых специализаций (ситуация 1). То есть, в той ситуации, которая считается одной из наиболее типичных при расширении кода [21]. Во-вторых, в большинстве других случаев (ситуации 3, 4, 5, 6) на практике, даже при использовании ОО языков, применяются методы процедурного и мультипарадигменного программирования. Ситуация 2 зачастую сглаживается тщательной проработкой интерфейса на этапе проектирования. Помимо этого во многих случаях легко можно пойти на небольшие изменения классов (ситуации 2, 4, 6), так как подобная модификация больше касается процесса компиляции, а не изменения модульной структуры программы. Можно пойти и на алгоритмические ухищрения (ситуации 3, 4, 7), которые не приводят к большому увеличению кода, но сохраняют требуемую гибкость, описанную в ситуации 1.

Таблица – Эволюционное расширение в простых ситуациях при использовании различных парадигм

Ситуация	**Подходы**		
	Процедурный	**ОО**	**ПП**
1 Добавление специализации и ее обработчиков	нет	есть	есть
2 Добавление процедур с дополнительной функциональностью	есть	нет	есть
3 Добавление новых полей в существующий тип	косвенное, для расширяемых типов	косвенное, при наличии RTTI	косвенное, при использовании обобщенной записи
4 Добавление обработчика специализации, включенной в обобщение	есть	нет	есть процедурный и параметрический
5 Создание обобщения на основе существующих специализаций	есть	косвенное	есть
6 Добавление в программу мультиметода	есть	нет	есть
7 Изменение мультиметода при добавлении специализации	нет	нет	есть

Процедурно-параметрическое программирование (ППП) обеспечивает эволюционное расширение программы практически во всех ситуациях. В ряде случаев можно использовать альтернативные решения, опирающиеся на чистый процедурный подход или на процедурно-параметрический полиморфизм. Последнее может повысить безопасность разработки программ за счет применения полиморфизма вместо явной проверки типов. Анализ простых ситуаций показывает, что ППП обладает гибкостью, перекрывающей процедурный и ОО подходы. Представляет интерес использование возможностей ППП и в более сложных случаях, образуемых комбинациями простых ситуаций.

ЛИТЕРАТУРА

1 Легалов, А.И. Разнорукое программирование // А.И. Легалов – http://www.softcraft.ru/paradigm/dhp/index.shtml.

2 Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования.: Пер. с англ. / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес – Спб.: «Питер», 2001. – 368 с.

3 Легалов А.И. Процедурно-параметрическая парадигма программирования. Возможна ли альтернатива объектно-ориентированному стилю? – Красноярск: 2000. Деп. рук. № 622-В00 Деп. в ВИНИТИ 13.03.2000. - 43 с.

4 Легалов, А. И. Процедурный язык с поддержкой эволюционного проектирования. / А. И. Легалов, Д. А. Швец // Научный вестник НГТУ. – 2003. – № 2 (15). – С. 25-38.

5 Легалов, И. А. Применение обобщенных записей в процедурно-параметрическом языке программирования. / И. А. Легалов // Научный вестник НГТУ. – 2007. – № 3 (28). – С. 25-38.

6 Керниган, Б.У. Язык программирования С. 2-е изд.: Пер с англ / Б.У Керниган, Д.М. Ритчи – М.: Издательский дом «Вильямс», 2009. – 304 с.

7 Павловская, Т.А. Паскаль. Программирование на языке высокого уровня: Учебник для вузов. / Т.А. Павловская - Спб.: «Питер», 2007. – 393 с.

8 Wirth, N. Programming in Oberon. A derivative of Programming in Modula-2 (1982). / N. Wirth – http://www.oberon.ethz.ch/wirthPiO/.

9 Moessenboeck, H. The Programming Language Oberon-2. / H. Moessenboeck, N. Wirth – Institut fur Computersysteme, ETH Zurich July. – 1996.

10 Сайт компании Oberon Microsystems, посвященный языку программирования Component Pascal – http://www.oberon.ch/

11 Barnes, J. Programming in Ada 95, 2nd Edition. / J. Barnes – Addison-Wesley. – 1998.

12 Страуструп, Б. Язык программирования С++. Третье издание.: Пер. с англ. / Б. Страуструп – СПб.; М.: «Невский диалект» – «Издательство БИНОМ», 1999. – 991 с.

13 Троелсен, Э. Язык программирования C# 2010 и платформа .NET 4.0. 5-е изд.: Пер. с англ. / Э. Троелсен – М.: ООО «ИД Вильямс», 2011. - 1392 с.

14 Эккель, Б. Философия Java. Библиотека программиста. 4-е изд.: Пер. с англ. / Б. Эккель – Спб.: «Питер», 2009. – 640 с.

15 Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++, 2-е изд.: Пер. с англ. / Г. Буч – М.: "Издательства Бином", СПб: "Невский диалект", 1998 г. - 560 с.

16 Элджер, Дж. С++: библиотека программиста.: Пер. с англ. / Дж. Элджер. – СПб.: ЗАО "Издательство Питер", 1999. – 320 с.

17 Легалов, А. И. ООП, мультиметоды и пирамидальная эволюция. / А. И. Легалов // Открытые системы. – 2002. – № 3. – С. 41-45.

18 Легалов, А. И. Мультиметоды и парадигмы. / А. И. Легалов // Открытые системы. – 2002. – № 5. – С. 33-37.

19 Александреску, А. Современное проектирование на С++: Пер. с англ. / А. Александреску – М.: Издательский дом «Вильямс», 2002. – 336 с.

20 Мейерс, С. Наиболее эффективное использование С++. 35 новых рекомендаций по улучшению ваших программ и проектов: Пер. с англ. / С. Мейерс – М.: ДМК Пресс, 2000. – 304 с.

21 Горбунов-Посадов, М. М. Расширяемые программы. / М. М. Горбунов-Посадов – М.: Полиптих, 1999.