ОСВІТА ТА ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 004.415.2 (043.3)

Национальный авиационный университет

Ю.М. Крамар

ПРИМЕНЕНИЕ СТИЛЕЙ ПРОГРАММИРОВАНИЯ В КОНСТРУИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В статье рассматриваются вопросы, рекомендуется ли обучать студентов бакалаврата «Программная инженерия» применению стилей программирования, в какую из набора специальных дисциплин учебного плана следует включить получение знаний по этой тематике и в каком объеме, какие при этом затрагивать аспекты и как прививать студентам практические навыки применения стилей.

У статті розглядаються питання, яким чином рекомендується навчати студентів бакалаврату «Програмна інженерія» застосуванню стилів програмування, в яку з набору спеціальних дисциплін учбового плану бакалаврату необхідно включити отримання знань щодо даної тематики і у якому об'ємі, яких аспектів дотримуватися та як прививати студентам практичні навички застосування стилів

The questions of how it's recommended to teach students of the "Software Engineering" Baccalaureate to put in practice programming styles, in which a set of special subjects of the curriculum should include the acquisition of knowledge on this subject and to what extent, what aspects to affect and how to inculcate students with practical skills, are considered at the article.

Ключевые слова: программное обеспечение, инженерия программного обеспечения, конструирование программного обеспечения, стили программирования, правила стиля программирования, средства форматирования кода.

Введение

Если сделать экскурс в историю развития программирования как области человеческой деятельности, то можно проследить, когда появилось и как развивалось понятие стиля программирования, каким образом возникали и сменяли друг друга культуры программирования в различные периоды, и какие при этом соответствующие им стили использовались в кодировании [1, 2, 3, 4].

Не смотря на то, что интуитивно выработанные правила и рекомендации по составлению программ использовались до эпохи структурного программирования [2], понятие стиля было введено в программирование именно в этот период и впервые упомянуто в трудах Б. Кернигана и Д. Риччи [3]. Тогда стили были представлены в виде рассуждений о том, как следует писать программы, чтобы они отвечали требованиям понимаемости текста и принципам структурного программирования, получившим широкое распространение, начиная с 70-х годов.

Авторы не формулируют определение стиля, не указывают требования, которым он должен соответствовать, не ограничивают программистов в выборе тех или иных правил и рекомендаций по написанию кода, составляющих стиль.

Они лишь обращают внимание на приемы и особенности использования языковых конструкций, составления и форматирования, указывая, что это позволяет сделать текст удобным для чтения и понимания, облегчает его изменение, доработку и повторное применение.

Следование таким рекомендациям с учетом решаемой задачи и используемых средств программирования стало принято называть хорошим стилем программиста или программирования [5, 6, 7].

Кроме Б. Кернигана и Д. Риччи такие правила начали предлагать и другие авторы стилей [8, 9, 10]. Правила встречаются в литературных и электронных источниках, а так же применены и могут быть распознаны в открытых кодах. Часть из них входит в несколько стилей, а часть встречается лишь в некоторых из них.

Многие стили малоизвестны и используются узко, поэтому могут быть найдены только при целенаправленном поиске и анализе

источников и кода, а некоторые получили широкую известность и распространение. Так, например, сегодня широко известны Венгерская нотация по составлению обозначений [11], стиль обозначений Camel часто используется по умолчанию в средах разработки программного обеспечения, востребованы правила стилей, предложенных Б. Керниганом, Э. Олмэном [3, 12].

На сегодняшний день описано и используется множество разнообразных стилей, отличающихся правилами в зависимости от применяемых языков программирования, решаемых задач, технологий и методологий программирования. Описание стилей часто носит формальный характер, что позволяет однозначно и четко трактовать и применять правила при кодировании.

Кроме того определенные стилевые правила автоматически поддерживаются редакторами сред программирования. Фактически, свойство программы отвечать определенному стилю стало не абстрактным и ощущаемым только на интуитивном уровне, а идентифицируемым и измеряемым.

Студенты бакалавраты «Компьютерные науки», обучаясь кодированию в дисциплинах «Основы программирования и алгоритмические языки», «Объектно-ориентированное программирование», так или иначе сталкиваются с правилами или ограничениями, накладываемыми на тексты составляемых ими на лабораторных работах программ.

Потому является желательным или даже необходимым включение и использование материала о стилях в теоретической части этих курсов, а также отработка навыков применения стилей в практической части.

Однако целью данной статьи является не подтверждение автором данной идеи, а определение роли и места обучения использованию стилей в подготовке студентов нового бакалавраты «Программная инженерия», в рамках которого студенты должны получить знания и навыки применения систематического, дисциплинированного, измеряемого подхода к разработке, использованию и сопровождению программного обеспечения (IEEE 1990) [13].

Поднимаются и обсуждаются вопросы: в какую дисциплину учебного плана бакалавраты можно включить получение знаний по этой тематике и в каком объеме, какие аспекты затрагивать при этом и какие практические навыки прививать студентам.

Анализ профессиональных и образовательных стандартов для определения места обучения применению стиля в бакалаврате «Программная инженерия»

Для того, чтобы определить, необходимо ли знакомить студентов бакалавраты «Программная инженерия» со стилями программирования и прививать навыки их использования при кодировании или использовании готовых исходных текстов программ, предлагается рассмотреть профессиональные и образовательные стандарты, на основе которых создавались учебный и рабочий учебный планы данного бакалавраты.

Если говорить об образовательном стандарте SE2004 [14], то в рекомендациях по преподаванию инженерии программного обеспечения в вузах можно найти место обучению применения стиля в вводных курсах CCCS CS101 «Основы программирования» и CS102 «Объектно-ориентированная парадигма», а также в базовом курсе по инженерии программного обеспечения SE211 «Конструирование программного обеспечения», так как данные курсы содержат освоение и совершенствование практики кодирования, которая может осуществляться с ограничениями, накладываемыми некоторым заданным стилем программирования, что имитирует определенную производственную обстановку и условия работы в команде разработчиков.

Если руководствоваться профессиональным стандартом SWEBOK [15], то изучая иерархию областей знаний, представленную в SWEBOK, можно найти ряд ссылок не только на само кодирование как область, непосредственно связанную с применением стилей программирования, но и на стандарты, в которых явно указано их использование. Подробнее остановимся на рассмотрении этой иерархии.

SWEBOK - документ, основной задачей которого является определение и систематизация тех аспектов деятельности, которые составляют суть профессии инженера-разработчика программного обеспечения, документ определяет 10 областей (рис.1). Они задают основы знаний процессов жизненного цикла программного обеспечения, инструментов, методов его создания и сопровождения, оценки продуктов фаз и управления. Одной из таких областей является конструирование программного обеспечения, включающее себя основы и управление конструированием, а так же практические соображения.

Основными позициями, указывающими на связь данной области знаний с аспектами применения стилей программирования, есть ниже следующее.

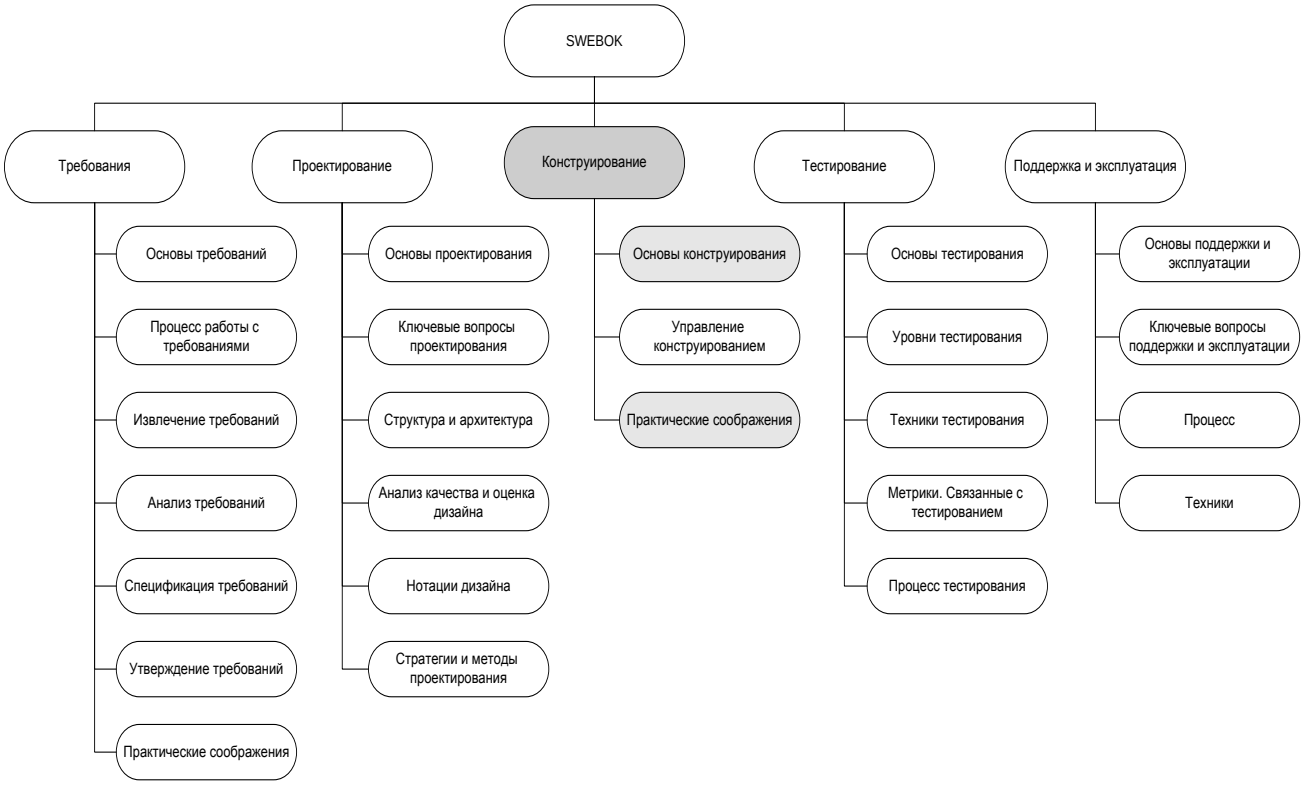

Рис. 1. Первые 5 областей знаний по SWEBOK

Во-первых, под конструированием понимается детальное создание рабочей программной системы посредством комбинации кодирования, верификации (проверки), модульного тестирования (unit testing), интеграционного тестирования и отладки [15]. То есть присутствуют процессы кодирования, в которых можно задействовать стили программирования.

Во-вторых, если изучать сущность составляющих область знания конструирование (рис.2), то в описании подобласти «Основы конструирования» можно найти определение фундаментальных основ конструирования программного обеспечения, таких как минимизация сложности (Minimizing Complexity) и стандарты в конструировании (Standards in Constructing).

Минимизация сложности в конструировании программного обеспечения достигается “при уделении особого внимания созданию простого и легко читаемого кода, пусть и в ущерб стремлению сделать его идеальным (например, с точки зрения гибкости или следования тем или иным представлениям о красоте, утончённости кода, ловкости тех или иных приемов, позволяющих его сократить в ущерб размерам и т.п.). Это не значит, что должно ущемляться применение тех или иных развитых языковых возможностей используемых средств программирования. Это подразумевает “лишь” придание большей значимости читаемости кода…” [15]. При этом указывается, что минимизация кода достигается, в частности, следованием стандартам (“Стандарты в конструировании”), использованием ряда специфических техник (“Кодирование”).

Стандарты, применяющиеся в конструировании, включают «языки программирования и соответствующие стили кодирования (например, Java Language Specification, являющийся частью стандартной документации JDK – Java Development Kit и Java Style Guide, предлагающий общий стиль кодирования для языка программирования Java)» [15]. То есть это говорит о том, что документом SWEBOK не только устанавливается косвенная связь между конструированием и использованием стилей, так как часть процессов конструирования есть кодирование, но в данном документе существует и прямое указание на стилевые правила и стандарты.

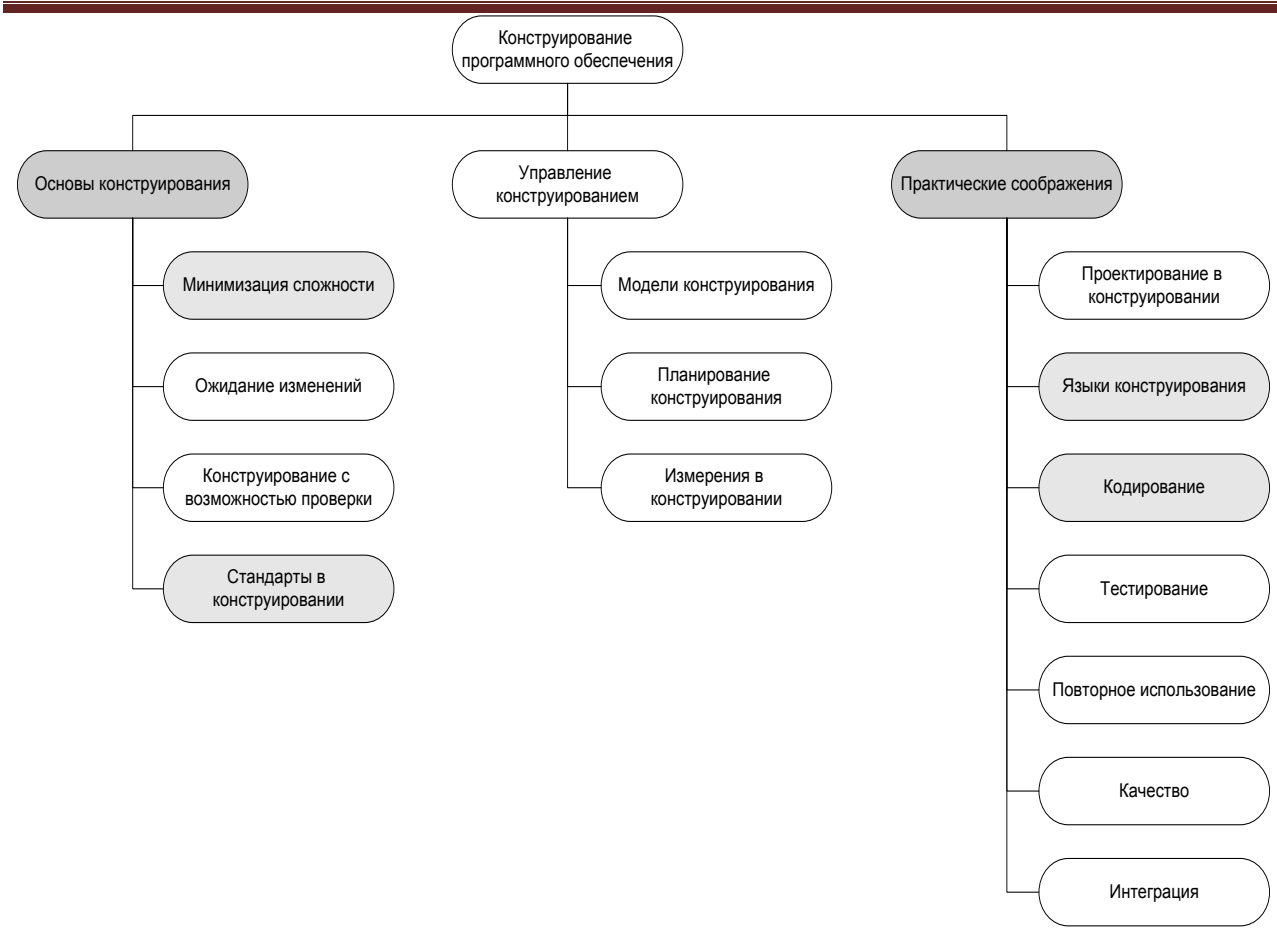

Рис. 2. Область знаний “Конструирование программного обеспечения” [SWEBOK, 2004, с.4-2]

Еще одна подобласть конструирования, которая может рассматриваться в связи со стилями программирования, это «Практические соображения» (рис. 2), и ее составляющие части «Языки конструирования» и «Кодирование».

Под языками конструирования понимаются все языки, позволяющие описать проблему и ее решение различными способами (вербально, графически и т.д.), и к ним относятся также языки программирования. Так как стиль программирования, отраженный в синтаксисе и программных конструкциях определенного языка программирования, становится стилем языка программирования [8, 9], то имеется непосредственная связь между изучением и использованием конкретных языков программирования и стилями.

Если рассматривать практику конструирования, то она показывает активное применение соображений и техник, в числе которых:

- техники создания легко понимаемого исходного кода на основе использования соглашений об именовании, форматирования и структурирования кода;
- использование классов, перечисляемых типов, переменных, именованных констант и других выразительных сущностей;
- организация исходного текста (в выражения, шаблоны, классы, пакеты/модули и другие структуры).

Очевидно, что здесь так же присутствует связь со стилями, так как соглашения об именовании и форматировании составляют большую часть стилевых правил программирования, применение которых позволяет составлять удобные для чтения, понимания и длительного использования программные тексты. А правила применения языковых конструкций и организации исходного текста программы являются дополнительными стилевыми ограничениями на синтаксис языка программирования. Соблюдение этих правил придает исходному тексту стиль программы, то есть свойство отвечать некоторому стилю программирования [8, 9].

Постановка проблемы

Рассмотрев профессиональные и образовательные стандарты, на основе которых создавались учебный и рабочий учебный планы

бакалаврата «Программная инженерия», можно прийти к выводу, что студенты данного направления обучения в рамках ряда образовательных дисциплин, в частности «Конструирования программного обеспечения», должны получить фундаментальные знания в области стилистики программирования и практические навыки, связанные с применением стилей на определенных этапах разработки программного обеспечения (в частности, в кодировании и конструировании).

Если данная необходимость установлена, то требуют уточнения конкретные аспекты обучения применению стилей программирования, которые предлагается рассмотреть в статье далее.

Использование стилей программирования в практике кодирования в аспекте конструирования программного обеспечения

Студенты бакалаврата «Программная инженерия» прежде чем приступить к изучению определяющих специальность инженера по разработке программного обеспечения ключевых дисциплин направления, касающихся специфицирования требований к программному обеспечению, проектирования, конструирования и тестирования программного обеспечения, оценки его качества и проектного менеджмента, на первом-втором курсах должны получить знания о языках программирования и навыки составления текстов программ. Для этого ими изучается ряд подготовительных дисциплин, так как овладевание языками и техниками программирования требует много усилий на изучение и наработку опыта для эффективного применения при решении конкретных задач [15].

Материал дисциплин «Основы программирования» и «Объектно-ориентированное программирование» должен быть построен и структурирован так, чтобы студенты изучили один или несколько языков программирования, причем, не привязываясь жестко к конкретному языку, а получив понимание операторного базиса, основных программных конструкций и правил их использования, изучив процедурное программирование, основные принципы объектно-ориентированной парадигмы, могли самостоятельно освоить и применять в кодировании любой из современных языков программирования.

При этом ими должно быть освоено структурное программирование, как одна из основных методологий, легших в основу стилей написания читаемых и понимаемых текстов программ, а также основы форматирования текста при изучении структурных операторов.

Кроме этого, студенты в рамках дисциплины «Групповая динамика и коммуникации» получают понимание того, что на сегодняшний день работа программиста и разработчика программного обеспечения носит коллективный характер, потому без использования стандартов и формальных договоренностей между членами команды любого проекта не обойтись.

Преподаватель также, как правило, накладывает собственные стилевые ограничения, требуя от студентов составлять и оформлять программы лабораторных работ определенным образом для выработки навыков применения стиля и, конечно, для упрощения работы коллектива «преподаватель-студенты».

Все это приводит к тому, что студентам прививается отношение к стилям программирования как производственной необходимости.

Что касается материала о стилистике программирования, излагаемого студентам в теоретической части курсов, то студентам можно в лекционный материал включить исторические сведения и обзор широко используемых стилей, существующих форм и источников описания стилевых правил, рассмотреть область применения различных стилей и выполнить со студентами сравнительный их анализ.

Анализируя касающиеся стилей программирования вопросы и темы для обсуждения на форумах разработчиков программного обеспечения, можно часто встретить просьбы начинающих программистов объяснить, где найти документы-описания, например, стиля Кернигана-Ритчи или стиля Олмэна. Это свидетельствует о том, что попадая в рабочую среду коллектива разработчиков определенной компании, выпускники сталкиваются с необходимостью писать код в соответствии с принятыми в ней требованиями, в том числе и стилевыми. И при условии, что студент, обучаясь кодированию и конструированию, ознакомлен с практикой применения стилей, у него быстро нарабатывается принятый в компании стиль программиста [16, 17]. При отсутствии подобных навыков студент вынужден обращаться за помощью к коллегам, а также часто на форумы, посвященные данной тематике.

Поэтому, если говорить о востребованных и известных программистам стилях кодирова-

ния, то в первую очередь выделяют стили 1TBS (One True Bracing Style - единственный правильный стиль расстановки скобок, еще называемый K&R (Kernighan and Ritchie) или kernel стилем), впервые описанный и использованный при составлении примеров кода книги "The C Progamming Language"; стиль Олмэна, впервые им применен в исходных кодах утилит для операционной системы BSD, поэтому часто называется "стиль BSD"; стиль Уайтсмита, использовавшийся в IDE компилятора Whitesmith C; стиль GNU, примененный в исходных текстах проекта GNU и др. Их сравнительные описания присутствуют в электронных источниках, в интернете. Однако, проанализировав их, можно сказать, что приведенные в них стили имеют узкую направленность, и правила стиля программирования имеют своей целью преимущественно задать вид и отступы структурных операторов (рис. 3).

Более полные и подробные описания стилей, состоящие из правил, направленных на различные аспекты текстов программ, представлены в специальных документах. Примеры подобных документов полезно представить студентам для изучения.

```
void fun(int par)  {
        if (par == 1)  {
                printf("It's 1TBS");
        }
}
```

```
void fun(int par)
        {
         if (par == 1)
                {
                 printf("It's Whitesmith");
                 }
        }
```

```
void fun(int par)
{
        if (par == 1)
        {
                printf("It's Allman");
        }
}
```

```
void fun(int par)
        {
                if (par == 1)
                {
                 printf("It's GNU");
                 }
        }
```

Рис.3. Особенности форматирования различных стилей программирования

Самым известным из них является описание «Indian Hill C Style and Coding Standards» и его модификациях «Recommended C Style and Coding Standards», 1990, 1997 гг. Данный документ был сформирован для AT&T's Indian Hill labs, чтобы выработать единый набор стандартов и рекомендаций кодирования разработчиков Indian Hill.

Авторами документов являются L.W. Cannon, R.A. Elliott, L.W. Kirchhoff, J.H. Miller, J.M. Milner, R.W. Mitze, E.P. Schan, N.O. Whittington (Bell Labs), Henry Spencer (Zoology Computer Systems, университет Торонто), David Keppel (EECS, UC Berkeley CS&E, университет Вашингтона) и Mark Brader (компания SoftQuad Incorporated, Торонто).

Целями использования стиля авторы отмечают способствование четкой и последовательной организации кода, повышающей степень его понимаемости, улучшение переносимости и сопровождаемости, уменьшение количества ошибок, допускаемых при написании кода.

Документ основывается на работах, в которых описаны синтаксис и семантика языка С, рекомендации по использованию языковых конструкций [18, 19], улучшению переносимости программного обеспечения [20, 21], применение специальных программных средств, например lint [22, 23], для оценки программ на предмет некорректного использования конструкций языка, ухудшающего такие характеристики кода как понимаемость, отказоустойчивость, переносимость.

Одними из основных трудов по описанию стилевых правил и рекомендаций, которые используют авторы, являются вышеупомянутые труды Б. Кернигана, Д. Риччи и П. Плагера [18, 19], в которых вводится понятие хорошего стиля в программировании, его назначение, приведено описание и пояснение отдельных правил стиля.

Авторами документа указывается, что не столько важно полностью принять весь перечень стандартов и обязательно следовать всему набору, сколько отобрать лишь необходимые из них в определенных условиях и быть последовательным в их применении, то есть выдерживать стиль во всем тексте программы. Только тогда использование стиля в кодировании будет эффективным.

В стандарте присутствуют соглашения об именовании и структуре файлов проекта, виде и расстановке комментариев, объявлении программных объектов, использовании отступов и пробелов, форме структурных операторов, составлении мнемонических обозначений, учитываются соображения переносимости и совместимости, правила форматирования, а также уделено внимание средствам проверки кода программного обеспечения таким как lint. То есть авторы документа советуют использовать выработанную ими опытным путем практику применения определенного набора стилевых правил, программируя на C, с использованием автоматизированных средств проверки «узких» частей кода.

Кроме рассмотренного документа для сравнения полезно ознакомиться с содержанием еще ряда стандартов, например, указанном в SWEBOK «Java Style Guide», 1998, компания Sun Microsystems, [24] или «GNU Coding Standards», Free Software Foundation [25]. То есть сущность изучения и применения студентами подобных документов такова, чтобы они имели представление о предмете стилистики программирования, осознавали важность и необходимость использования стилей при составлении программ, получили практические навыки составлять исходные тексты в соответствии с требованиями заданного стиля и умели при необходимости приобретать новый стиль как свойство программиста при изменении инструментальных средств разработки программного обеспечения и специфики решаемой задачи.

На сегодняшний день инженерия программного обеспечения как отрасль располагает широким набором средств и методов анализа, проектирования, интеграции и тестирования, управления проектом, рисками и качеством проектируемых систем [26, 27]. Не исключением является и стилистика программного обеспечения, потому студентов следует ознакомить и приучить использовать средства и инструменты, помогающие поддерживать стиль при кодировании.

Если говорить об инструментах, то, во-первых, многие среды разработки программного обеспечения имеют встроенные средства поддержки стиля. Например, KDevelop (свободная интегрированная среда разработки для UNIX-подобных операционных систем), распознает и позволяет использовать не только стиль GNU, как собственный стиль разработчика среды, но и стили Кернигана, Олмэна, Уайтмэна, а также задавать пользовательский стиль.

Во-вторых, существуют утилиты-форматировщики исходных текстов программ, которыми пользуются программисты как для корректировки собственных текстов так и обработки перед использованием в своих проектах сторонних открытых кодов. Отличаются они интерфейсом (консоль, использование bat-файлов, графическое приложение), принципом обработки файлов исходных текстов (on-line или локальный режим), языком программирования и типом приложений обрабатываемых текстов.

Из широкоизвестных программ-форматировщиков это lint [22, 23] и ident с интерфейсом командной строки, обрабатывающие тексты на C и C++; он-лайн-форматировщик PrettyPrinter (клиентская часть размещена по адресу http://www.prettyprinter.de/) для обработки текстов на PHP, Java, C++, C, Perl, JavaScript, CSS; Highlight (GUI-приложение размещено на http://www.andre-simon.de/doku/highlight/en/highlight_demo.html) для обработки кодов HTML, XHTML, RTF, LaTeX, TeX, SVG, BBCode с возможностью использования цвета шрифтов; консольный ArtisticStyle, обрабатывающий коды на C++ и C, Java, C#; многоязыковая утилита UniversalIndentGUI с графическим интерфейсом и др.

И наконец, так как инженерные методы, используемые в разработке программного обеспечения, часто основываются на измерениях, а количественными оценками являются его метрические характеристики, то применение стилей программирования также может быть связано с измерением определенных свойств программных текстов, отражающих примененные в них стили. Для этого возможно использование как ряда общеизвестных метрик (например, длина кода, степень его комментирования, цикломатическая сложность по Маккейбу), так и метрик, специально созданных для оценки характеристик текста программы, которые регламентируются пра-

вилами стилей (например, длина идентификатора, количество операторов безусловного перехода, количество литералов, количество параметров подпрограммы). И если в дополнение к этому в практической части курса студентами и преподавателем будут использоваться какие-либо программные средства [9], автоматизирующие эти измерения с целью контроля применения стилей при составлении программ на лабораторных работах, то при оценивании уровня знаний студента окажется возможным учесть не только знание предметов программирования или конструирования, но и его способность точно придерживаться заданного стиля.

Заключение

Анализ профессиональных и образовательных стандартов программной инженерии показал необходимость изложения теоретических и практических основ стилистики программирования при подготовке инженеров-разработчиков программного обеспечения. И привить студентам навыки применения стилей программирования в кодировании предлагается в рамках освоения ими процессов конструирования программного обеспечения.

Список литературы

1. *Ван Тассел Д.* Стиль, разработка, эффективность, отладка и испытание программ: Пер. с англ. – М.: Мир, 1980.
2. *Боровин Г. К.* Ошибки-ловушки при программировании на Фортране. – М.: Наука, 1987. – 144 с.
3. *Керниган Б., Ритчи Д.* Язык программирования Си: Пер. с англ., 3-е изд., испр. - СПб.: "Невский Диалект", 2001. - 352 с.
4. *Sidorov N.A. Software stylistics.* // Вісник НАУ. -№2. -2005. – С.98-103
5. Good Programming Style [Электронный ресурс]- режим доступа: http://www.eg.bucknell.edu/~xmeng/Course/CS2330/Handout/StyleKP.html
6. Programming Style Joseph Bonneau [Электронный ресурс]- режим доступа: http://www.jbonneau.com/style_guide.pdf
7. Toward Developing Good Programming Style C version (McCann) [Электронный ресурс]- режим доступа: http://www.cs.arizona.edu/~mccann/style_c.html
8. *Мейерс С.* Эффективное использование C++. 50 рекомендаций по улучшению программ и проектов: Пер. с англ. – М.: ДМК, 2000. – 240 с.
9. *Ален И. Голуб. С и С++.* Правила программирования: Пер. с англ. – М.: БИНОМ. – 272 с.
10. *Нуквист Е.* Правила хорошего тона для программирования на С++: Пер. с англ. – К.: Наук. думка, 1994. – 85 с.
11. *Charles Petzold* Programming Windows, 5th Edition. Microsoft Press.- November, vol. 11. - 1998. - 1479 p.
12. The Jnly Correct Indent Style [Электронный ресурс]- режим доступа: http://www.terminally-incoherent.com/blog/2009/04/10/the-only-correct-indent-style/
13. IEEE Standards Association [Электронный ресурс]- режим доступа: http://standards.ieee.org/findstds/standard/
14. Software Engineering 2004. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. A Volume of the Computing Curricula Series IEEE Computer Society, Association for Computing Machinery, 2004 [SE, 2004].
15. Guide to the Software Engineering Body of Knowledge (SWEBOK), IEEE Computer Society 2004 Version [SWEBOK, 2004].
16. *Сидоров М.О.* Стилистика программирования // Проблеми інформатизації та управління: збірник наукових праць. – 2003. - Вип. 8. – С. 204-207.
17. *Крамар Ю. М.* Автоматизация решения задач стилистики программирования // Проблеми інформатизації та управління: Зб. наук. пр.: вип. 5. – К.: НАУ, 2002. – С. 211-215.
18. *B.W. Kernighan and D.M. Ritchie*, The C Programming Language, Prentice Hall 1978, Second Ed, 1988.
19. *Brian W. Kernighan and P. J. Plauger* The Elements of Programming Style. McGraw-Hill, 1974.
20. *B.A. Tague, C* Language Portability, Sept 22, 1977.
21. *J. E. Lapin* Portable C and UNIX System Programming, Prentice Hall 1987.
22. *S.C. Johnson*, Lint, a C Program Checker, USENIX UNIX Supplementary Documents, 1986 [Электронный ресурс] - режим доступа: http://www.chris-lott.org/resources/cstyle/indhill-cstyle.pdf
23. *Ian F. Darwin*, Checking C Programs with lint, O'Reilly & Media, 1988. – 81 p.
24. Java Coding Style Guide, Achut Reddy, Server Management Tools Group, Sun Microsystems, Inc., 1998 [Электронный ресурс] - режим доступа: http://developers.sun.com/sunstudio/products/archive/whitepapers/java-style.pdf
25. *GNU Coding Standards, Richard Stallman*, Free Software Foundation [Электронный ресурс]- режим доступа: http://www.gnu.org/prep/standards/
26. *Лаврищева Е.М., Петрухин В.А* Методы и средства инженерии программного обеспечения: учебное пособие. – М.:МФТИ (ГУ), 2006.
27. *Иан Соммервилл,* Инженерия программного обеспечения, 6е издание, М: издательство «Диалектика», 2006. - 624 с.

Сведения про автора

Крамар Юлия Михайловна - к.т.н., доцент кафедры инженерии программного обеспечения Национального авиационного университета, научное направление – инженерия программного обеспечения, стилистика программирования.
E-mail: yulia.kramar@livenau.net

Статья поступила в редакцію 12.02.2011 г.