УДК 004.413:338.5

СТИЛИСТИКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Н.А. Сидоров

Национальный авиационный университет
03058, Киев, проспект космонавта Комарова,1, т.: 406 7396;
E-mail:sna@nau.edu.ua

Considered is the application of style in software development. Stylistics of software as a section of the software engineering is introduced.

Рассматривается приложение стиля к разработке программного обеспечения. Введено понятие стилистики программного обеспечения как области инженерии программного обеспечения.

Введение

В связи с распространением инженерных методов разработки программного обеспечения (ПО), моделей жизненного цикла, основанных на компонентной разработке и повторном использовании [1, 2], и экстремального программирования [3] ставятся и решаются задачи, связанные с чтением текстов программ, написанных на разных языках программирования и в разное время [4 – 8]. Известно, что на характер текста программы влияют не только алгоритм или язык программирования, но и идеологические, культурные особенности того периода времени, в котором создавалась программа [8]. Например, на рис.1 показан текст работающей программы, на языке Си (взято из Internet), которая написана специальным „запутанным" способом. Не зная идеи, которую автор положил в основу способа написания программы нужно затратить очень много времени на ее понимание. Поэтому, чтобы понимать тексты программ, часто необходимо знать или указанные особенности периода времени их написания, или господствующие в этом периоде идеологии и идеи авторов. Следовательно, повышается ценность таких программ, которые не только работают, но и некоторым систематическим образом отражают распространенные в периоде времени их написания идеологию, корпоративную идею или культурные особенности. Это ведет к тому, что программисту необходимо уметь представлять идею или идеологию и передавать представление вместе с программой. В различных областях этот аспект деятельности человека относится к стилю, а его исследование – является предметом стилистики [9]. Расширение применения инженерных методов в ПО показывает, что стиль в разработке ПО также, как и в другой человеческой деятельности, можно связывать не только с программными текстами, но и с другими продуктами фаз жизненного цикла, например, архитектурой [10].

```
#include <stdio.h>
 main(t,_,a)
char *a;
{return t>1?t<3?main(-79,-13,a+main(-87,1-_,
main(-86, 0, a+1 )+a)):1,t<_?main(t+1, _, a ):3,main ( -94, -27+t, a
)&&t == 2 ?_<13 ?main ( 2, _+1, "%s %d %d\n" ):9:16:t<0?t<-72?main(_,
t,"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#{l,+,/n{n+\
,/+#n+,/#;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l q#'+d'K#!/\
+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n')(){}#}w'){){nl]'/+#n';d}rw' i;# ){n\
l]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#\
n'wk nw' iwk{KK{nl]!/w{%'l##w#' i; :{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;\
#'rdq#w! nr'/ ') }+}{rl#'{n' ')# }'+}##(!!/'')
:t<-50?_==*a?putchar(a[31]):main(-65,_,a+1):main((*a == '/')+t,_,a\
+1 ):0<t?main ( 2, 2 , "%s"):*a=='/'||main(0,main(-61,*a, "!ek;dc \
i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-O;m .vpbks,fxntdCeghiry"),a+1);}
```

Рис. 1. Текст "запутанной" программы

Применение стиля в разработке ПО - давняя, но систематически неисследованная проблема. Первые результаты по исследованию стиля были изложены в [11]. Результаты следующих исследований представлены в [12, 13]. И.В. Вельбицкий вводит графический стиль программирования [14]. А.П. Ершов пишет о стиле, как о фундаментальном профессиональном свойстве программиста, отмечая роль учебного процесса в приобретении свойства стиля и роль производственных требований в его сохранении [15]. И.В. Поттосин описывает требования, которым должна удовлетворять «хорошая» программа [16]. В работах [17-20] отражены результаты исследований современных аспектов стиля, связанных, в основном с использованием конструкций объектно-ориентированных языков программирования.

Работа состоит из четырех частей. В первой части вводятся основные понятия стилистики ПО и на примере программирования эти понятия конкретизируются. Во второй части рассматриваются особенности

эпох (культур) программирования и приводятся характеристики идей и стилей имеющих место в каждой эпохе. Третья часть посвящена архитектурному стилю, в ней приводится краткая характеристика наиболее распространенных архитектурных стилей. В четвертой части рассматриваются средства, которые используются для решения некоторых задач стилистики ПО.

1. Онтология стилистики программирования

Определим стилистику программирования, как раздел инженерии программного обеспечения [1, 2] предметом изучения которого является применения стиля в программировании.

Анализ литературы [4 – 8], [10 – 20] показывает, что в программировании нет удовлетворительного определения стиля. Нет такого определения и в других областях деятельности человека, что бы его можно было использовать в программировании. Поэтому, следуя основным положениям работы [9], за основу рассуждений о стиле возьмем определение стиля как средства выражения некоторой идеологии или идеи в человеческой деятельности. Практика показывает – стиль имеет широкое распространение, если его формирование происходит в течение некоторого времени, под влиянием объективно складывающейся идеологии, как правило, усилиями многих субъектов человеческой деятельности. Вместе с тем, стиль может быть и отражением идеи одного субъекта или группы субъектов (часто в рамках одного вида деятельности), и тогда – это личный или корпоративный стиль, как правило, не распространяющийся широко. Чем масштабнее идеология или идея, тем разнообразнее человеческая деятельность, в которую они проникают посредством стиля. И наоборот, чем менее масштабны и более субъективны идеология или идея, тем скорее стиль можно ассоциировать с приемом или методом человеческой деятельности.

Идеология, как предпосылка для появления стиля в некоторой области человеческой деятельности может возникнуть в этой области, или вне нее. Например, революционная коммунистическая идеология породила множество стилей в разных областях человеческой деятельности [9]; перестройка российского общества привела к изменению публицистического стиля [21]; идея, суть которой состоит в том, что наиболее естественной для человека формой представления информации является графический образ, привела к разработке языков, в которых программы рисуются [14]; идея экономить вычислительные ресурсы вследствие отсутствия мощных средств выполнения вычислений, в свое время, привела к появлению не только соответствующего стиля написания программ [13], но и архитектурного стиля построения процессоров. Вместе с тем, например, выражение идеи упрощения процесса понимания программ путем написания идентификаторов, используя символ-префикс как обозначение типа значений (Венгерская нотация [22]), скорее прием, чем стиль.

Таким образом, рассматривая стиль, следует принимать во внимание два измерения: одно отражает множество идеологий и идей, а другое – множество видов человеческой деятельности. Определяя стиль человеческой деятельности, прежде необходимо идентифицировать идеологию или идею, которую он выражает, а затем, проецируя их на человеческую деятельность, определить другие понятия, связанные с ним в этой деятельности.

Очевидно, определяя стиль, который нашел применение в разных областях человеческой деятельности достаточно описания характерных черт или признаков соответствующей идеологии или идеи. Тогда это описание будет представлять стиль как доменно-независимое понятие. Рассматривая стиль с онтологических позиций, как объект («сущность», «вещь»), обладающий свойствами необходимо указать существенные свойства стиля и его связь с другими объектами домена [23, 24]. Будем определять стиль - «сущность» (класс) как систему трех следующих свойств (рис. 2):

- выражать некоторую идеологию или идею;
- иметь период (время) существования;
- иметь связь с человеческой деятельностью.

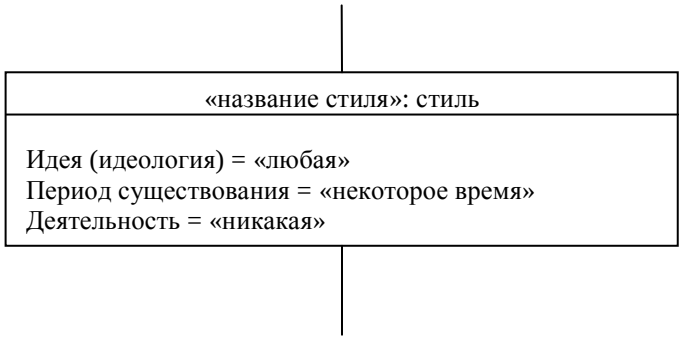

Рис. 2. Класс – стиль

Для стиля как доменно-зависимого понятия все три свойства являются существенными [24]. Первое и второе свойства остаются существенным всегда, т.е. являются качествами стиля как доменно-независимого

понятия. Существенность третьего свойства ведет к доменно-зависимому понятию стиля человеческой деятельности.

Таким образом, стилю как доменно-независимому понятию соответствует сущность (средство) выражающая какой-то период времени, некоторую идеологию или идею, способом, не связанным с конкретной человеческой деятельностью. По сути стиль представляет основу, на которой строятся стили различных человеческих деятельностей. Поэтому на диаграмме рис. 2 свойство «идея» имеет значение «любая», свойство «деятельность» – «никакая», а период существования – «некоторое время».

Рассматривая стилистику программирования как предметную область (домен), для представления её онтологии будем использовать как компьютерный так и математический подходы [25]. Применение первого подхода будет демонстрироваться с помощью UML-диаграмм, а для второго используем аксиоматический метод, распространенный при описании предметных областей баз данных [26]. Представление стиля в рамках второго подхода может иметь вид St = <A, S, D>, где A – множество собственных аксиом стиля не зависящих от существа выражаемой идеологии, S и D – множество аксиом, описывающих характерные черты идеологии стиля в статике и динамике. Последние множества могут использоваться и для описания стиля программирования.

Множество A может состоять из следующих аксиом:

1) единственность стиля: если есть идеология (I) и на ней основан стиль (St), то больше не существует стилей, основанных на этой идеологии

$$\forall I St(I) \sim (\bar{S}t(I) = St(I));$$

2) существование стиля человеческой деятельности: если есть идеология (I), основанный на ней стиль St, и человеческая деятельность P, то существует и стиль человеческой деятельности $St_p(St(I), P)$ основанный на стиле St(I)

$$\forall I St(I) \sim \forall P St_P(St(I), P);$$

3) рефлективность: любой стиль есть подстиль самого себя

$$\forall St(St = St);$$

4) антисимметричность: подстиль (стиль, построенный на некотором стиле) не может быть стилем для образовавшего его стиля

$$\forall St_1 \forall St_2 (R(St_1, St_2) \sim \neg R(St_2, St_1));$$

5) транзитивность: если стиль (St_2) есть подстиль некоторого стиля (St_1) и стиль (St_3) есть подстиль стиля (St_2), то он есть подстиль стиля (St_1)

$$\forall St_1, \forall St_2, \forall St_3 (R(St_1, St_2) \sim (R(St_2, St_3) \sim R(St_1, St_3))).$$

Аксиомы статики S описывают особенности состояния стиля в конкретном домене человеческой деятельности. Обычно в определении стиля участвует несколько сущностей и отношений между ними. Описание их состояний и описывает статику. Аксиомы динамики D описывают изменения, которые происходят в периоде существования стиля. Например, аксиома описывающая свойство «период существования стиля»: если есть стиль St, то он обладает периодом существования T, которое имеет начало и конец:

$$\forall\ StT(St, (t_1,t_2)) \sim (t_1 < t_2).$$

Для описания динамики изменений в домене можно использовать модальную логику [26].

Давая значение свойству «человеческая деятельность» в классе стиль рис. 3 можно получать стиль той или иной человеческой деятельности.

Рис. 3. Класс – стиль программирования

Таким образом, стиль программирования – это стиль, используемый в деятельности человека (домене), сущность которой состоит в написании программ для компьютеров.

Рассматривая в человеческой деятельности три сущности (субъект, инструмент, продукт), и приняв, что в программировании такими сущностями являются программист, язык программирования и программа, можно построить модель домена программирование рис. 4.

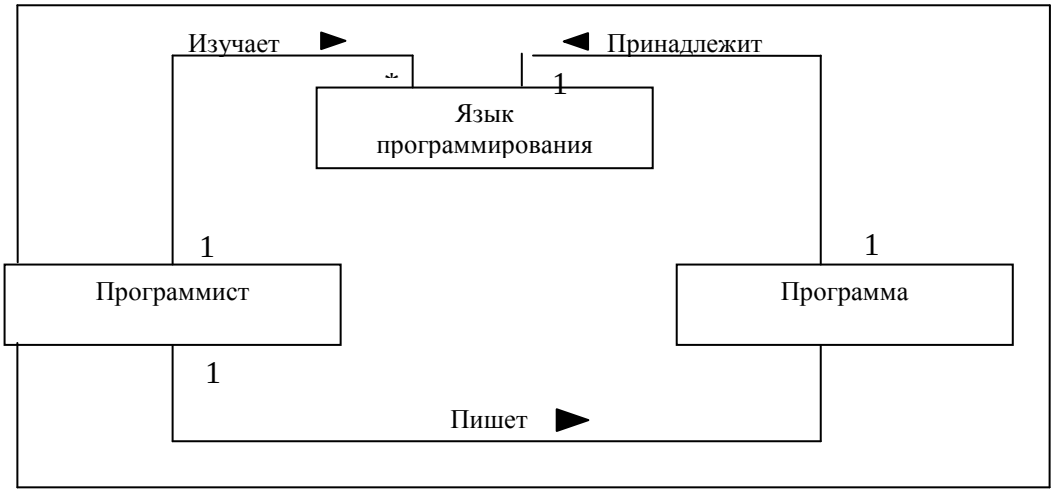

Рис. 4. Домен программирование

Рассматривая связь указанных сущностей со стилем человеческой деятельности, можно определить следующие понятия: стиль субъекта, стиль инструмента и стиль продукта. Для домена программирование им соответствуют следующие понятия: стиль программиста, стиль языка программирования и стиль программы. При этом, если субъект и продукт от связи со стилем приобретают дополнительные свойства (стиль субъекта-программиста и продукта-программы) рис. 5, то инструмент участвует в создании новой сущности (стиль инструмента-языка программирования) рис. 6.

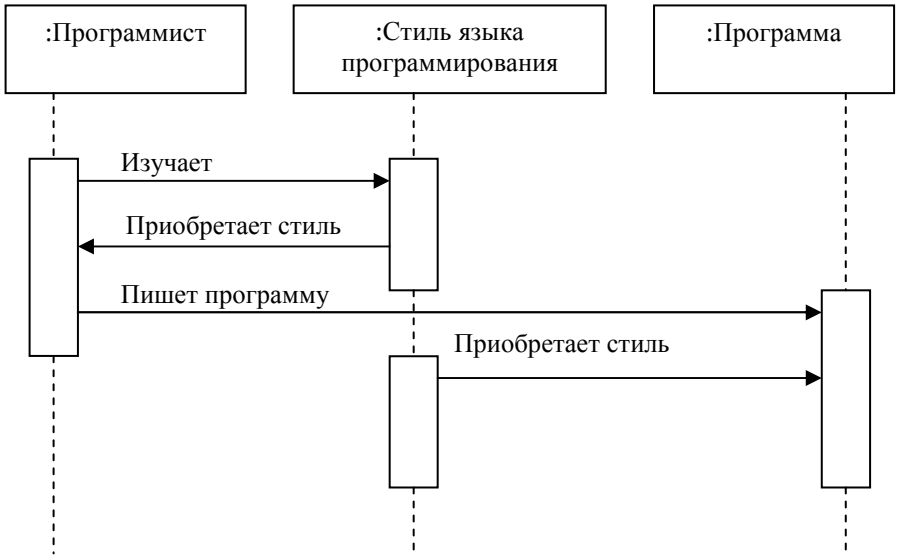

Рис. 5. Диаграмма последовательностей

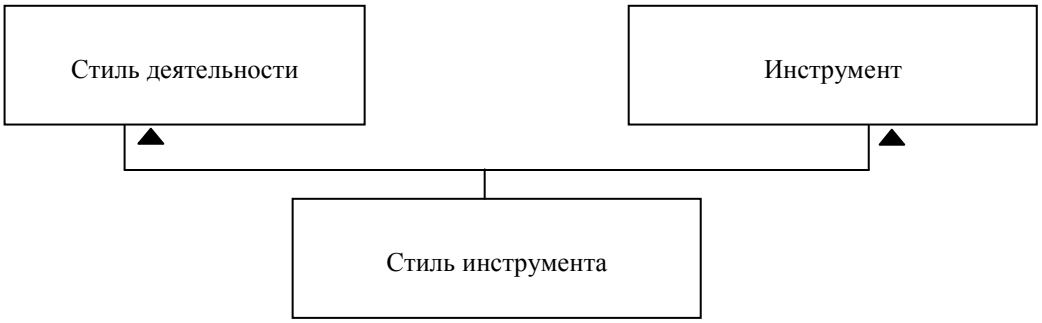

Рис. 6. Стиль инструмента

Фрагмент модели домена программирование с учетом влияния на него стиля представляется тремя классами – стиль языка программирования, программист и программа (рис. 7). Стиль языка программирования – это средство (подмножество языка программирования), в котором обеспечена поддержка определенного стиля программирования (рис. 8). В стиле языка программирования обычно реализуется некоторое подмножество стиля программирования. Поддержка должна быть в виде средств, обеспечивающих представление соответствующего стиля программирования в рамках средств языка программирования. Как правило, выражение стиля программирования не выходит за рамки лексики и синтаксиса языка программирования и обеспечивается автоматически. Однако стиль программирования должен точно выдерживаться в программах, поэтому в стиле языка программирования необходимо предусмотреть средства, обеспечивающие контроль применения стиля программирования, например, подобно тому, как контролируется синтаксис. Это обеспечит обязательную реализацию свойства стиля программы в контексте данного стиля языка программирования. Реализовать контроль можно путем применения эмпирических методов и средств, в частности измерений и измерителей [27, 28].

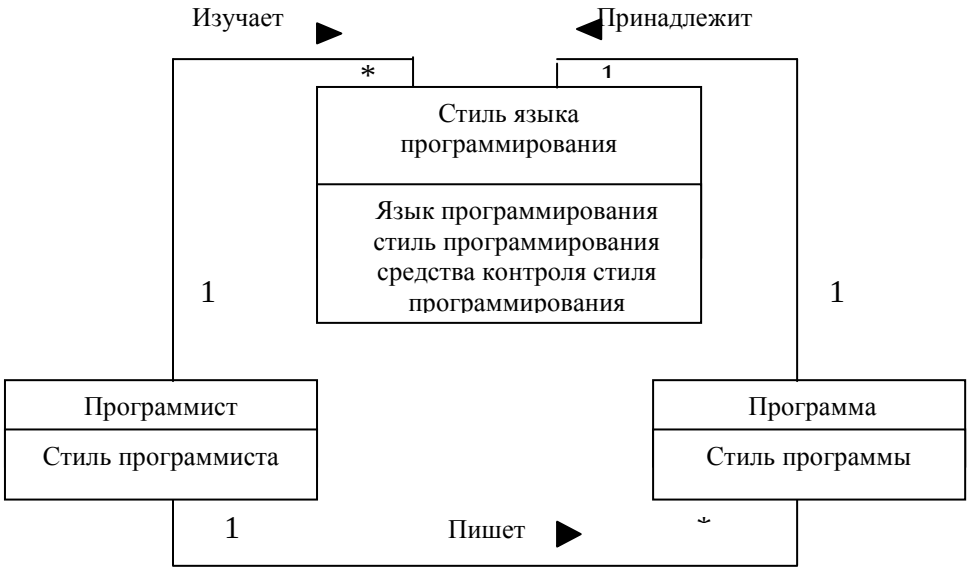

Рис. 7. Домен программирование с учетом стиля

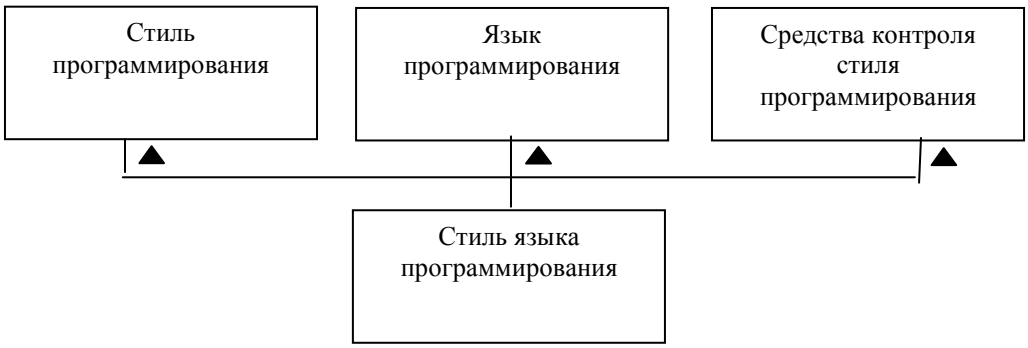

Рис. 8. Стиль языка программирования

Стиль программиста – это приобретенное свойство программиста знать определенный стиль программирования и применять его при написании программ. Приобретение этого свойства осуществляется программистом путем изучения стиля языка программирования. Строго говоря, стиль программиста должен быть его профессиональным свойством, например, таким, как для художника умение рисовать, и должен приобретаться в процессе профессионального обучения [15]. Со стилем программиста связаны процессы мотивации обучения, переобучения и применения стиля программирования.

Стиль программы – это свойство программы удовлетворять требованиям стиля языка программирования (принадлежать к множеству программ стиля языка программирования). Неполное удовлетворение требований ведет к стилизации программы. Например, программисты часто используют стилизации под Венгерскую нотацию.

Таким образом, основным системо-образующим фактором стиля является идеология или идея. Стиль может складываться какое-то время, а затем дифференцироваться в человеческих деятельностях. Какой-то период времени может складываться и господствовать один стиль – «стиль эпохи» [9].

2. Стиль в программировании

За время существования программирования сменилось несколько «эпох» [8, 29]. В каждой устанавливалась и господствовала своя к ультура, которая обладала идеологией определяющей характерный для эпохи стиль программирования. Будем различать четыре эпохи (табл.1), определив их отношением к одному из фундаментальных явлений в программировании, структурному программированию [30], непосредственно связанному с написанием текстов программ, а значит и стилями программирования.

Эпохи программирования Таблица 1

Эпохи программирования	Характеристика парадигмы				
	Период	Ориентация	Идеология	Внимание	Методы и стили
«до структурного программирования»	1951 - 1975	На процессор, исполнителя программ	Эффек-тивность программ	Технике исполнения программ	Загадочное программирование, программирование прозой, шаблонное программирова-ние
«структурного программирования»	1975 - 1990	На человека-программиста, читателя программ	Понимае-мость	Технике программи-рования	Структурное программирование, «понятное» программирование
«после структурного программирования»	1990 - 1996	На человека-проектиров-щика программ	Многора-зовое и повторное использо-вание	Технике решения задач, проекти-рованию	Модульное и объектно-ориентированное программирова-ние
«инженерии программного обеспечения»	1996	На человека-инженера по ПО	Реализация ПО в заданных условиях	Доказательному, удовлетворяемому предъявляемым требованиям программи-рованию	Эмпирическое программирование, литературное программирование

Эпоха «до структурного программирования». В ней компьютеры были медленные, а память ограниченной и все идеи направлялись на построение программ, эффективных по быстродействию и памяти. Техника программирования независимо от языка ориентировалась на особенности архитектуры компьютера и, следовательно, на исполнителя программ. Программисты писали коротко, хорошо используя особенности систем команд компьютеров, возможности операционных систем и языков программирования. Такое программирование называют загадочным [31]. К основным идеям, влияющим на стиль, можно отнести идеи краткости и эффективности. Поэтому было широко распространено нестандартное использование систем команд процессоров и операторов языков программирования, создание самомодифицирующихся программ, использование общих областей памяти, повсеместное использование оператора go to. Отсутствие одной идеи программирования было почвой для существования разных стилей программирования. Например, имело место шаблонное программирование, яркими представителями средств, обеспечивающих его были генераторы отчетов, или программирование прозой, для реализации которого использовался COBOL [30].

Эпоха «структурного программирования». В ней был разработан метод структурного программирования. Идеи связывались с построением читабельных программ. Техника программирования ориентировалась на программиста-читателя [6]. Основное внимание уделялось языку программирования и методам, которые способствуют написанию понятных программ. Вопросы эффективности реализации программ переносятся на уровень компиляторов. Широкое распространение структурного программирования и его основной идеи – понимаемости, привели к становлению стиля программирования, который ориентирован исключительно на написание структурно понятных программ [7].

В качестве примера рассмотрим описание структурного стиля программирования, применение которого даст структурную программу. Для простоты описания не будем рассматривать свойства, связанные с оптимизацией программы и аксиомы динамики. Кроме аксиом A стиль будет описываться следующими аксиомами S [30, 32, 33]:

1. Операторный базис (структурные операторы) структурной программы:

если S – это оператор присваивания, P и P_R^l – пост и предусловия, то

$$\forall S \exists P \exists P_R^l (P_R^l \sim \omega p(S, P));$$

если S – это оператор выбора, P – постусловие, S_i^0 – охраняемый оператор, B_I – охрана, то

$$\forall S \exists P ((\exists i : 1 \le i \le n : B_i) \wedge$$
$$(\forall i : 1 \le i \le n : B_i \sim \omega p(S_i^0, P)) \sim \omega p(S, P));$$

если S – это оператор повторения, $H_k(P)$ – слабейшее предусловие, то

$$\forall S \exists P ((\exists k : 0 \le k : H_k(P)) \sim \omega p(S, P)).$$

2. Структурная программа:

если операторы S_i – это структурные операторы i = 1….n, то структурная программа P_S – это последовательность структурных операторов

$$\forall S_i P(S_1 S_2 ... S_n);$$

3. Целевая направленность структурной программы: пусть есть последовательность состояний программы [32, 33] A_0, A_1, …A_m, где $A_{i+1} = S_i(A_i)$, S_i – структурный оператор, выполняемый на i-ом шаге, тогда программа имеет целевую направленность, если $\forall A_i (A_{i+1} \ne A_i)$ или для каждого оператора S_i с предусловием Q должно быть истинным $\neg(Q \sim \omega p(S_i, Q))$.

Если текст программы основывается на приведенных аксиомах, то он будет написан структурным стилем программирования [30].

Эпоха «после структурного программирования». Развитие структур данных, абстрактных типов данных и модульного программирования привело к тому, что основные идеи программирования стали направляться на построение многократно используемых программ. Получили развитие методы генерационного и композиционного программирования [34]. Основная идея этих методов состояла в том, чтобы создавать программы из готовых блоков, либо вставляя их в готовые архитектуры, либо связывая, используя операторы языков в качестве «клея». Методами, используемыми для программирования таких блоков стали модульное и объектно-ориентированное программирование. В связи с реинженирингом наследуемого ПО широкое практическое применение находит повторное использование [2]. Объектом внимания программиста становится класс (модуль) как абстрактный тип данных, а техника создания программ ориентируется на решение задач. Программист из читателя стал проектировщиком, широко использующим накопленный опыт. Основное внимание уделяется методам решения задач, объектно-ориентированному анализу и проектированию. На операторном уровне по-прежнему используется идея понимаемости.

Эпоха «инженерии программного обеспечения». Труд разработчиков ПО приобретает инженерный характер. Программист из одиночки, занимающегося искусством программирования, становится инженером, работающим в коллективе, который создает и сопровождает программные продукты [34]. Основные идеи направляются на разработку ПО в заданных условиях и требованиях. Акцент деятельности программиста переносится на доказательное удовлетворение сформулированных требований, значительную роль играет метрическая теория ПО [35]. При этом основным методом становится эмпирическое программирование, а его реализацией, в частности литературное программирование, сущность которого состоит в том, что текст программы встраивается в текст документации [4]. Наряду со стилем программирования важную роль начинает играть архитектурный стиль [10].

3. Стиль в программном обеспечении

Таким образом, посредством архитектур понятие стиля стало глубже проникать в ПО и возникло понятие архитектурного стиля.

Архитектура – это представление основных структурных, функциональных и потребительских свойств ПО [10]. Обычно, архитектура составляется из компонентов двух типов – вычислительных (клиенты, сервера, фильтры, уровни, базы данных) и соединительных (вызовы, события, протоколы, трубы). Взаимодействуя между собой, компоненты образуют архитектуру ПО.

Архитектурный стиль – это средство выражения некоторой идеологии или идеи в виде архитектурной модели. Поэтому архитектурный стиль определяет семейство архитектур ПО, представляя перечень вычислительных, соединительных компонентов и набор правил, указывающих то, как они образуют ту или иную архитектуру. Существующие в настоящее время архитектурные стили и их характеристики приведены в табл. 2.

Архитектурные стили Таблица 2

Стиль	Примеры архитектур	Тип программных систем
Потоковый	Пакетная – последовательная, труба и фильтр	Ориентированные на обработку потоков данных
Вызовов и возвратов	Главная программа – подпрограммы, объектно-ориентированная, иерархических уровней	Объектно-ориентированное
Независимых компонентов	Процессо - ориентированная, События - ориентированная	Событий и процессов
Виртуальных машин	Интерпретаторы, правило - базированная	Виртуальных машин
Репозитарно-ориентированный	Базы данных, гипертекстовая система, классная доска	На основе баз данных и репозитариев
Клиент-серверный	Распределенная	Клиент-серверные

Потоковый стиль представляет идею, что каждый вычислительный компонент имеет вход и выход, а соединительный компонент является потоком данных. Соединяя первые с помощью вторых получаем некоторую «вычислительную» топологию, которая направляется на решение тех или иных задач.

Стиль вызовов и возвратов представляет идею, что каждый вычислительный компонент является объектом, который взаимодействует с другими объектами путем осуществления вызовов и возвратов. В объектно-ориентированной архитектуре объекты являются абстрактными типами данных, а соединительные компоненты – сообщениями.

Стиль независимых компонентов представляет идею, что вычислительный компонент объявляет одно или несколько событий, которые регистрируются и все зарегистрированные события запускаются.

Стиль виртуальных машин представляет идею, что ПО моделирует некоторую аппаратную часть вычислительной системы.

Стиль репозитарно-ориентированный представляет идею, что ПО состоит из двух компонентов – центральная структура данных (репозитарий), которая представляет текущее состояние и совокупность независимых компонентов, которые оперируют на центральной структуре данных.

Стиль клиент-серверный представляет идею, что ПО составляется из двух типов компонентов – клиентов и серверов, последние обслуживают клиентов.

Применение стиля в ПО, будем рассматривать в контексте жизненного цикла, который состоит из следующих фаз [36, 37]:

- формирование стиля под влиянием идеологии или идеи;
- идентификация стиля (определение и представление характеристических признаков идеологии или идеи);
- создание стиля программирования на основе идентифицированного стиля;
- создание стилей языков программирования, поддерживающих стили программирования или создание архитектурных стилей;
- обучение программистов и архитекторов («создание» стилей программистов и архитекторов);
- применение стилей языков программирования при написании программ («создание» стилей программ) и архитектурных стилей при архитектурном проектировании («создание стилей программных систем»);
- «отмирание» стиля, как правило, вследствие «отмирания» идеологии или идеи.

Если стиль определяется для выражения «естественной» (объективной) идеологии, то фазы жизненного цикла могут перекрываться во времени. При этом, как показывает опыт тот стиль, который идентифицируется, никогда полностью не реализуется, а на его основе создается множество более «простых», «практичных» стилей, таким образом происходит дифференциация стиля [9]. В противном случае сначала формируется идея, затем стиль и далее, фазы выполняются последовательно во времени. В жизненном цикле стиля программирования, так же как в жизненном цикле ПО [2], можно различать процессы, ресурсы и продукты, которые в настоящее время плохо изучены.

Таким образом, стилистика ПО – это раздел инженерии ПО [1, 2], изучающий следующие концепции: стиль программирования и архитектурный стиль, как средства выражения идеологии или идеи в человеческой деятельности – разработке ПО; стиль программиста, как профессиональное свойство, обеспечивающее применение стилей программирования; стили языков программирования и других инструментов, как средств для реализации стилей программирования; стили программ – свойства программ – как результат применения стилей языков программирования. Цель стилистики ПО направлена на изучение указанных концепций на основе жизненного цикла стиля ПО, в контексте жизненного цикла ПО, путем изучения процессов, ресурсов и продуктов фаз жизненного цикла ПО и построения соответствующих технологий, обеспечивающих его реализацию.

4. Ресурсы стилистики программного обеспечения

Ресурсы стилистики ПО – это методы и средства, которые используются при реализации процессов жизненного цикла стиля ПО. В настоящее время ресурсы, в основном, применяются для реализации процессов

использования стилей программирования. Их можно разделить на три группы методов и средств автоматизирующих следующие процессы [38, 39]:

- создание программ, отвечающих стилю программирования;
- проверку текстов программ на соответствие стиля;
- преобразование текстов программ из одного стиля в другой;

К средствам первой группы относятся редакторы исходных текстов, которые делятся на два типа: редактор исходных текстов сред языков программирования [38] и редакторы визуального программирования [39].

Средства второй группы осуществляют проверку текстов готовых программ (например Lint) . Обычно проверяется стиль программирования «зашитый» в средстве.

Средства третьей группы («красивая печать») осуществляют преобразование текста готовой программы в текст, отвечающий стилю программирования, описание которого представляется в командной строке средства (например, cb, indent, astyb).

Средства применения архитектурных стилей:

- проектирования (выбор архитектурного стиля и архитектуры, выбор шаблонов для реализации архитектуры);
- описания (языки описания архитектур программ обеспечения).

Комплексный (горизонтальный в контексте жизненного цикла) подход к использованию методов и построению средств стилистики ПО развивается в работах [36, 37, 40–43]. Сущность его состоит в том, что бы реализовать некоторую информационную технологию, направленную на поддержку всех фаз жизненного цикла стиля программирования.

Заключение

В работе [44] А.П. Ершов, рассматривая внутреннюю природу и эстетическую сущность программирования указывает, что профессия программиста «приближается к писательскому делу», а разработка и сопровождение ПО напоминает то, что произошло в результате появления книгопечатания, а именно подобно книгам, накапливающим внешний образ мира глазами авторов, программы накапливают информационные и операционные модели мира. Как в первом, так и во втором случае стиль и его применение в программировании является, на наш взгляд ярким выражением подмеченных А.П. Ершовым особенностей профессии программиста или говоря современным языком – разработчика (инженера) ПО. Поэтому стилистика ПО заслуживает большего, чем ей уделялось до настоящего времени, внимания программистской общественности.

1. *Соммервил И.* Инженерия программного обеспечения. – М.: Вильямс, 2002. – 620 с.
2. *Сидоров Н.* Повторное использование, переработка и восстановление программного обеспечения // –УсиМ. 2000. – № 3.
3. *Glass R.L.* Extrime programming: the good, the bad, and the bottom line. – IEEE Software. – 2001. – 18. – 6. – p. 111-112.
4. *Knuth D.E.* Literate programming. – Computer J. – 1984. – v. 27. – № 2.
5. *Weisen M.* Source Code. – Computer. – 1987. – P. 66 - 73
6. Goldberg A. Programmer as Reader. - IEEE Software. – 1987. – P. 62 - 70.
7. *Robson D.J, Bennett K.H., B.J. Cornelins B.J., Munro M.* Approaches to Program Comprehension. –Systems Software. – 1991. - 14, № 2. - P. 79 – 84.
8. *Railich V., Wilde N. et. al.* Software cultures and evolution Computer. – 2001, Sept. - P. 25 – 28.
9. *Соколов А.Н.* Теория стиля. – М.: 1968. – 210 с.
10. *Bosch I.* Design and use of software architectures. – Addison – Wesley. – 2000. – 325 p.
11. *Тассел В.* Стиль, разработка, эффективность, отладка и испытание программ. – М.: Мир, 1980. – С. 280.
12. *Керниган Б., Плоджер Ф.* Элементы стиля программирования. – М.: Радио и связь, 1984. – С. 160.
13. *Боровин Г.К.* Ошибки – ловушки при программировании на Фортране. – М.: Наука, 1987. – 144 с.
14. *Вельбицкий И.В.* Технология программирования. – Киев: Техніка, 1984. – 274 р.
15. *Ершов А.П.* Предварительные соображения о лексиконе программирования // Кибернетика и вычислительная техника. – М.: Наука, 1984.
16. *Поттосин. И. В.* «Хорошая программа»: попытка точного определения понятия Программирование. – 1997. – 2. - С. 3–17.
17. *Cargill T.* C++programming style. –Adtdison –Wesley. –1992.
18. *Очков В.Ф., Пухничев Ю.В.* 128 советов начинающему программисту. - М.:Энергоатамиздит, 1992. - 257 с.
19. *Нуквист Е.* Правила хорошего тона для программирования на С++. – Киев: Наук. думка, 1994. – 85 с.
20. *Мейерс С.* Эфективное использование С++. –ДМК. – М.: 2000.
21. *Миневич Г.Д.* Язык и политика: о длине предложения и ее колебаниях в современных русских газетных текстах. – НТП., М.: – 2000. Сер. 2. – № 8. – С. 1–7.
22. *Lammers S.* Programmers at work. – Microsoft Press. – Redmond Wash. – 1986
23. *Chandrasekaran, B. Josephson J.R., Benjamins V.R.* What are ontology's, and why do we need them? // IEEE Intelligent systems, 1999. – № 2 – P. 20–26.
24. *Уемов Ф.* Вещь, свойства, отношение. – М.: 1958.
25. *Клещев А.С., Артемьева И.А.* Математические модели онтологий предметных областей. ч. 1. Существующие подходы к определению понятия «онтология». – НТИ. – Сер.2. – Инф. процессы и системы. – 2001. – № 2. – С. 20 – 27.
26. *Андон Ф.И., Яшунин А.Е., Резниченко В.А.* Логические модели интеллектуальных информационных систем. – Киев: Наук. думка, 1999. – 396 с.
27. *Torii K., Matenmoto K.-I.* Ginger – 2: An Environment for Computer – Aided empirical Software engineering. – IEEE Transactions on Software engineering. 1999. – 25, № 4. – P. 474– 492.
28. *Сидоров Н.А., Хоменко В.А.* Структура средств для исследования зависимостей между метриками программного обеспечения // Проблеми підвищення ефективності інфраструктури.– НАУ. – К.: 2003, вип.10. – С. 222 – 225.

29. *Wiegers K.E.* Creating a software engineering culture. - Dorset House Publish. – 1996. – 350 p.
30. *Дал У., Дейкстера Э., Хоор К.* Структурное программирование. – М.: – Мир, 1975. – 240 с.
31. *Holmes N.* The Case for perspicuous programming. – Computer.– 2003. – 30, 4. – P. 102 – 104.
32. *Дейкстра Э.* Дисциплина программирования. – М.: Мир, 1978. – 276 с.
33. *Грис Д.* Наука программирования. – М.: Мир, 1984. – 411 с.
34. *Shaw M., Garlan D.* Software engineering – perspectives on an emerging discipline. –Printce Hall. – 1996.
35. *Pflegger D.* Software measurements. – Addison Welly 2000. – 535 p.
36. *Сидоров Н.А.* Стилистика программирования // Проблеми інформатизації та управління. –НАУ. – К.: – 2003, вип. 8. - С. 204 – 207.
37. *Крамар Ю.М., Сидоров Н.А.* Стили программирования и фазы жизненного цикла программного обеспечения ”Проблеми інформатизації та управління”. – К.: НАУ. – 2002. в.6. - С. 235– 237.
38. *Архангельский А.Я.* C++BUILDER 5. Интегрированная среда разработки. – М.: «Бином», 2000.
39. *Архангельский А.Я.* Интегрированная среда разработки Delphi 1– 5. – М.: «Бином», 2000. – 256 с.
40. *Крамар Ю.М.* Схемы систематизации правил стилів программирования // Проблемы программирования. Материалы 3-ей междунар. научно-практ. конф. По программированию «УкрПРОГ2002». Спец. Вып. – 2002. – № 1– 2. – С. 127 – 130.
41. *Крамар Ю.М.* Автоматизация решения задач стилистики программирования // Проблеми інформатизації та управління: К.: НАУ, 2002. – вип. 5. – С. 211 - 215.
42. *Крамар Ю.М.* Средства для автоматизированного синтеза стилей программирования // Вестник НАУ. – 2002. – №2 (13). – С. 52– 60.
43. *Крамар Ю.М.* Автоматизация процесса контроля применения стиля языка программирования // Проблемы программирования. Материалы Междунар. научно-практ. конф. по программированию «УкрПРОГ2004». Спец. вып. – 2004. – № 2-3. – С. 208 – 214.
44. *Ершов А.П.* О человеческом и эстетическом факторах в программировании // Кибернетика. – 1972. – № 5. – С. 95– 99.