UDC 004.4.(045)

N.A. Sidorov, Dr.Sci. (Eng.)

SOFTWARE STYLISTICS

Institute of computer technologies, NAU, 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.

Introduction

In connection with distribution of engineering methods of the software development, the models of life cycle, based on component development and reuse [1; 2], and extreme programming are put and solved the problems connected with reading of program texts, written on different programming languages and at various times [3–7]. It is known, that character of the program text is influenced not only algorithm or the programming language, but also ideological, and cultural features of the time in which the program was created [7]. Therefore, to understand texts of programs, it is often necessary to know either specified features of the time of their writing, or ideologies dominating over this period and ideas of authors. It leads to the necessity for a programmer to be able to represent idea or ideology and to transfer representation together with the program. In various areas, this aspect of activity of the person concerns style, and its research – is a subject of stylistics [8]. Expansion of application of engineering methods in software shows, that style in software development as in other human activity, can be connected not only with program texts, but also with other products of software life cycle phases, for example, architecture [9].

Researches and publications

Application of style in software development is old, but not often investigated problem. The first results of research style have been presented in work [10]. The results of following researches are presented in works [11; 12]. I.V.Velbitsky has introduced graphic style of programming [13] A.P. Ershov considered style as fundamental professional property of the programmer, marking the role of educational process in purchase of style property and the role of industrial requirements in its preservation [14]. I.V. Pottosin describes requirements which the "good" program should satisfy [15]. Works [16; 17] reflect the results of researches of modern aspects of the style which are mainly connected, with the use of designs of object-oriented programming languages.

Ontology of programming stylistics

Let's define stylistics of programming as a section of the software engineering [1; 2] which subject is application of style in programming. The analysis of literature [3–7; 9–19] shows, that in programming there is no satisfactory definition of style. Likewise, there exists no definition of other spheres of human activity, that could be used in programming. As a basis of reasoning on style we shall take the definition of style as means of expression of some ideology or an idea in human activity [8].

Thus, considering style, it is necessary to consider two measurements: one reflects a set of ideologies and ideas, and the other is a set of types of human activity. Defining style of human activity, first of all it is necessary to identify ideology or idea which it expresses, and then, projecting them on human activity to define other concepts connected with it in this activity. Obviously, defining style, which has found application in different areas of human activity, description of characteristic features or attributes of corresponding ideology or idea is enough. Then this description will represent style as a domain-independent concept. Considering style from ontological positions, as object ("essence", "thing"), possessing properties it is necessary to specify essential properties of style and its communication with other objects of the domain. We shall define style – "essence" (class) as a system of three following properties (fig. 1):

– to express some ideology or idea;
– to have the period (time) of existence;
– to have connection with human activity.

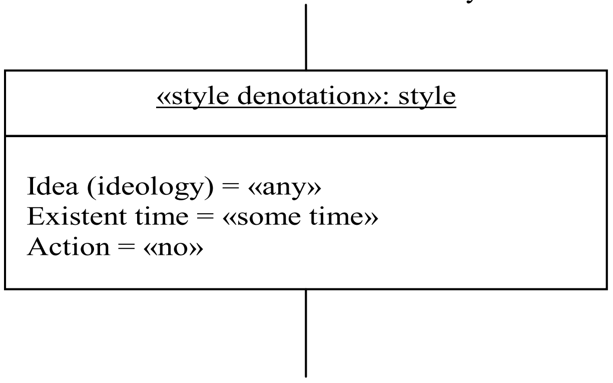

Fig. 1. Class-style

For style as domain-dependent concept all three properties are essential. The first and second properties remain essential always, that are qualities of style as domain-independent concept. Importance of the third property leads to the domain-dependent concept of style of human activity.

Thus, to style, as to domain-independent concept, there corresponds essence (means) expressing any period of time, some ideology or idea, the way which has not been connected with specific human activity. As a matter of fact, style represents the basis on which styles of various human activities are constructed. Considering stylistics of programming as a subject domain, for representation of its ontology, we shall use both computer and mathematical approaches [18]. Application of the first approach will be shown by means of UML-diagrams, and for the second we will use the axiomatic method widespread at the description of subject domains of databases. Representation of style within the limits of the second approach can look like St = <A, S, D>, where A – set of own axioms of style not depending on an essence of expressed ideology, S and D – sets of the axioms describing characteristic features of ideology of style in static and dynamics. The last sets can be used for description of style of programming.

Set A may contain such axioms:

1) uniqueness of style: if exists ideology I and style, based on it, then there are no styles that are also based on this ideology

$\forall ISt(I) \sim (\overline{St}(I) = St(I));$

2) existence of style of human activity: if there exists ideology I, style St, based on it, and human activity P, then exists style of human activity St_p(St (I), P), based on the style St(I)

$\forall ISt(I) \sim \forall PSt_p(St(I), P);$

3) reflexivity: every style is the substyle of itself

$\forall St(St = St)$;

4) antisymmetry: substyle (style, which is derived from some style) can't be a style for a style, it was based on it

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

5) transitivity: if style St_2 is substyle of some style St_1 and style St_3 is substyle of St_2, then St_3 is substyle of St_1

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

Axioms of static S describe peculiarities of condition of the style in definite domain of human activity. Usually in description of style, multiple essences and relations between them are used. Description of their conditions is description of static. Axioms of dynamics D describe changes that occur during style existence time. For example, axiom that describes property, is called "time of style existence": if exists style St, then it exists during some finite period of time:

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

For description of dynamics of changes in domain, it is possible to use modal logic. We may obtain different classes of human activity by assigning different values to the property "human activity" in class "style" (fig. 2).

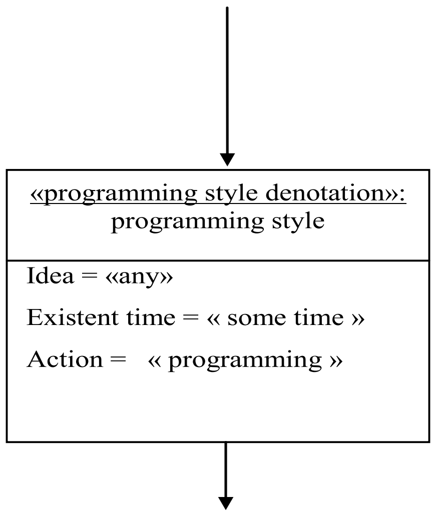
«programming style denotation»:
programming style
Idea = «any»
Existent time = « some time »
Action = « programming »

Fig. 2. Class-programming style

Thus, programming style is the style that is used in human activity (domain), whose essence consists in programming.

We are able to construct a model of the "programming" domain while considering in human activity (domain) three essences (subject, tool and product), and taking into consideration that in programming such essences are programmer, programming language and program (fig. 3).

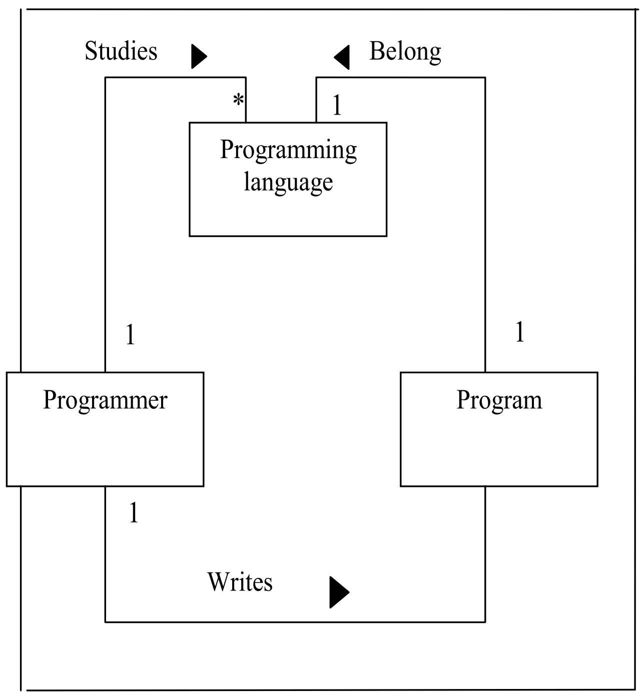

Fig. 3. Programming domain

Considering connection of essences, mentioned above, with style of human activity, we may define such conceptions, as style of a subject, style of a tool and style of a product. For "programming" domain, they correspond to such conceptions, as: programmer's style, style of programming language and style of program. At the same time, while subject and product obtain new properties because of connection to style (style of subject-programmer and product-program) (fig. 4), tool is involved into creation of new essence (style of tool-programming language)(fig. 5).

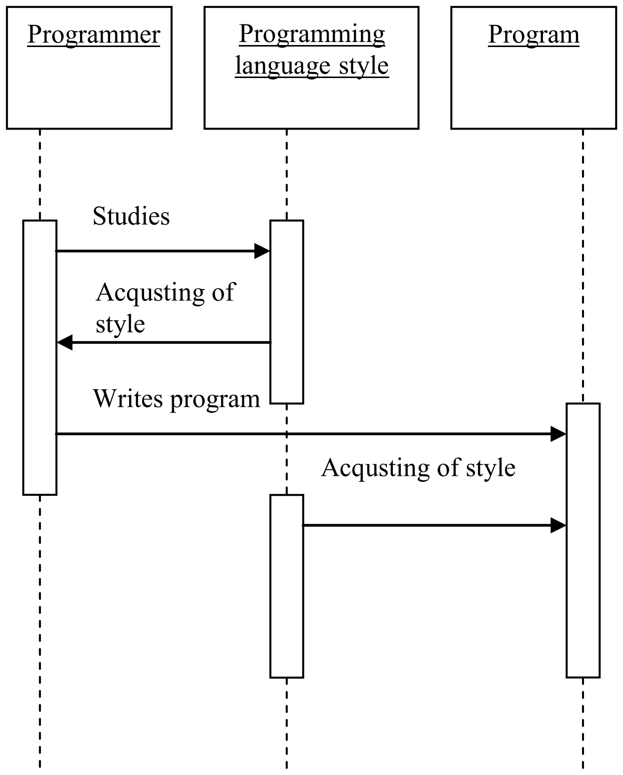

Fig. 4 Sequencer diagram

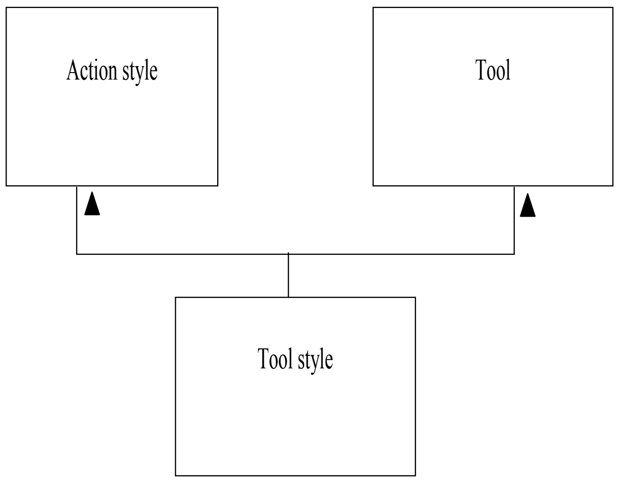

Fig. 5. Tool style

A fragment of model of "programming" domain, that takes into consideration influence of style, is represented by three classes – style of programming language, programmer and program (fig. 6).

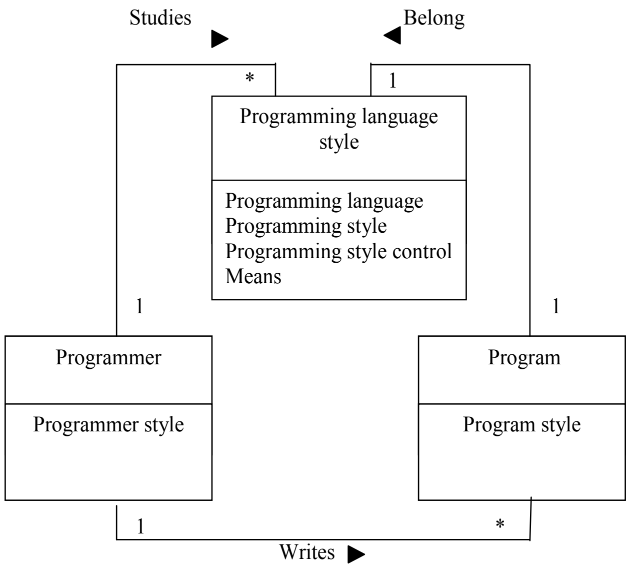

Fig. 6 Programming domain and style

Style of programming language is a tool (a subset of programming language), in which definite style of programming is supported (fig. 7).

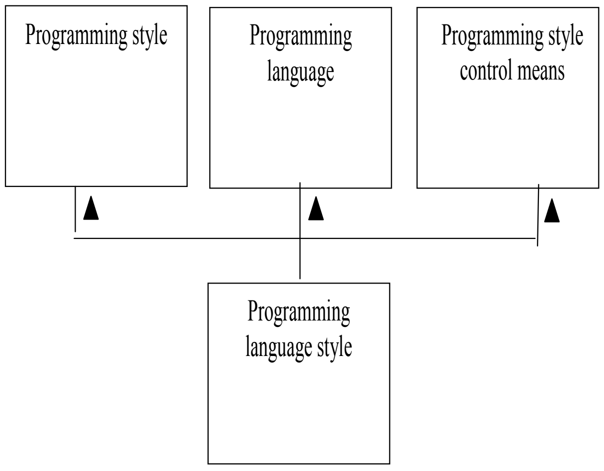

Fig. 7. Style of programming language

Usually, in style of programming language, some subset of programming style is realized. Support must be realized in the form of means, providing representation of corresponding style in the context of programming language means. As a rule, programming style expression doesn't exceed the limits of vocabulary and syntax of programming language and is provided automatically. But it is necessary to keep to style of programming, that's why it's necessary to provide tools, controlling programming style, just as the way controlling syntax. It will provide obligatory realization of style property of the program in the context of the given style of a programming language. It is possible to realize control over the programming style by means of empirical methods and tools, in particular – measurements and measuring devices.

Style of programmer is the acquired quality of programmer to be acquainted with definite style of programming and apply it in programming. Acquirement of this quality is performed by studying the style of programming language. In the strict sense, style of programmer must be his professional quality, like the ability to paint for an artist and must be acquired while training skills [14]. Style of programmer is connected with processes of motivation of training, retraining and application of programming style.

Style of a program is the ability of a program to satisfy requirements of the programming language. Incomplete satisfaction of the requirements of programming language leads to stylization of a program. For example, many programmers use stylization to Hungary notation.

Thus, the main system-generative factor for style is ideology or idea. Style may be formed during some time, and then differentiate in human activities. During some period of time, one style may arise and dominate – "style of the epoch".

Style in programming

During existence of programming, some epochs passed. In each epoch there was one culture that aroused and dominated and which had own ideology, defined style of programming typical for that epoch.

We may distinguish four epochs (tab. 1), using their relation to one of fundamental conception programming, structured programming, directly connected to code writing, so – with styles of programming.

Style in software engineering

By means of architectures, the notion of style began to penetrate deeper into software and so the notion of architectural style appeared.

Architecture is a conception of the main structural, functional and consumer properties of the software [9]. Usually architecture is composed of components of two types – computing (clients, servers, filters, levels, databases) and connecting (calls, events, protocols, pipes). Interacting with each other, these components form the software architecture.

Architectural style is a means of expressing some ideology or idea in the form of an architectural model or template. That's why the architectural stylistics designates the family of a program systems offering a list of computing and connecting components and a set of rules, that define how thy can be connected into architecture. The existing architectural styles and their characteristics are shown in tab. 2. Application of style in software, let us consider it in the context of software lifecycle, which consists of the following phases:

– formation, under the influence of ideology or an idea;
– identification of a style (determining and presenting characteristic features of the ideology or an idea);
– creating a programming style on the basis of the identified style;
– creating programming language styles, that support programming styles or creation of architectural styles;
– training programmers and architects ("creating" programmer and architect styles);
– applying programming language styles in the process of software writing ("creating" software styles) and architecture styles at the stage of architectural programming ("creating software system styles");
– style withers away, as a rule, because of "withering away" of the style's ideology.

The software stylistics is a part of software engineering [1; 2], which at present studies following conceptions:

Table 1

Programming Epochs

Programming Epochs	Paradigm facilities				
	Time	Orientation	Ideology	Attention	Methods and style
«Until structuring programming»	1951– 1975	On processor, program performer	Efficiency	Programming technique	Enigmatic programming, literate programming, template programming
«Structural programming»	1975– 1990	On programmer, reader of program	Understanding	Programming technique	Structural programming, understanding programming
«After structural programming»	1990–1996	On designer	Reusable	Design technique	Module and object oriented-programming
«Software engineering»	1996	On software engineer	Software design condition	Prove programming	Empirical, literate programming

Table 2

Architecture styles

Style	Architectures	Program system types
Dataflow	Batch, sequential, pipes and filters	Dataflow processing systems
Call-and-return- systems	Main program and subroutines	Object- oriented systems
Independent components	Communicating processes	Event systems
Virtual machines	Interpreters, Rule-based	Rule-based systems
Data-centered	Data bases, hypertext	Hypertext systems
Client-server	Distribution	Client-server systems

programming style and architectural style, as a means of expressing ideology or an idea in human activity – software development; programmer's style, as a professional quality, facilitating the use of programming styles; styles of programming languages and other tools, as the means of software style implementing; program styles – program properties as a result of software styles application. The goals pursued by software stylistic are directed towards studying the described concepts on the basis of software style lifecycle, in the context of software lifecycle, by studding the processes, resources and products of software lifecycle phases and developing technologies that facilitate their implementation.

Conclusion

A.P. Yershov, considering the inner nature and the aesthetical nature of programming, points out, that the profession of a programmer "approaches to the level of a writers profession", while development and support of the software is closer to that of a typography, namely in a similar way that the books accumulate the outer image of the world in the authors eyes, the programs accumulate informational and operational models of the world. In our opinion, in the first case and the second, the style and its application in programming is a striking example of the peculiarities of the programmer's profession noted by A.P. Yershov or in modern speech a software developer (engineer). That's why the software stylistics deserves more attention, than it has received from programmers.

Literature

1. *Соммервил И.* Инженерия программного обеспечения. – М.: Вильямс, 2002. – 620 с.
2. *Сидоров Н.* Повторное использование, переработка и восстановление программного обеспечения. – УсиМ. – 1998. – №3. – С. 74–84.
3. *Knuth D.E.* Literate are programming // Computer Journal. – 1984. – Vol. 27, N 2. – P. 42–44.
4. *Weisen M.* Source Code // Computer. – 1987. Nov. – P. 66–73.
5. *Goldberg A.* Programmer as Reader.-IEEE Software. – 1987. Sept. – P. 62–70.
6. *Robson D.J., Bennett K.H., Cornelins B.J., Munro M.* Approaches to Program Comprehension // J.Systems Software. – 1991. – Vol. 14, N. 2. – P. 79–84.
7. *Software* cultures and evolution // V. Railich, N. Wilde, M. Buckellew et. al. Computer. – 2001. – Sept. – P. 25–28.
8. *Соколов А.Н.* Теория стиля. – М., 1968. – 210 с.
9. *Bosch I.* Design and use of software architectures. – Addison–Wesley, 2000. – 325 p.
10. *Тассел В.* Стиль, разработка, эффективность, отладка и испытание программ. – М.: Мир, 1980. – 280 с.
11. *Керниган Б., Плоджер Ф.* Элементы стиля программирования. – М.: Радио и связь, 1984. – 160 с.
12. *Боровин Г.К.* Ошибки – ловушки при программировании на Фортране. – М.: Наука, 1987. – 144 с.
13. *Вельбицкий И.В.* Технология программирования. – К.: Техніка, 1984. – 274 р.
14. *Ершов А.П.* Предварительные соображения о лексиконе программирования //Кибернетика и вычислительная техника. – М.: Наука, 1985. – Вып. 1. – С. 199–210.
15. *Поттосин И.В.* «Хорошая программа»: попытка точного определения понятия // Программирование. – 1997. – 2. – С. 3–17.
16. *Нуквист Е.* Правила хорошего тона для программирования на С++. – К.: Наук. думка. –1994. – 85 с.
17. *Мейерс С.* Эффективное использование С++. – М.:ДМК. – 2000. – 325 с.
18. *Клещев А.С., Артемьева И.А.* Математические модели онтологий предметных областей. Ч. 1. Существующие подходы к определению понятия «онтология» // НТИ. Сер. 2. Информационные процессы и системы. – 2001. – №2. – С. 20–27.

The editors received the article on 8 July 2005.

М.О. Сидоров
Стилістика програмного забезпечення
Розглянуто застосування стилю в розробці програмного забезпечення. Запропоновано стилістику програмного забезпечення як розділ інженерії програмного забезпечення.

Н.А. Сидоров
Стилистика программного обеспечения
Рассмотрено применение стиля в разработке программного обеспечения. Введена стилистика программного обеспечения как раздел инженерии программного обеспечения.