КРИТИКА
И СЕМИОТИКА

Стихи в программном коде: современный опыт и методика анализа [*]

Б. В. Орехов

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ», МОСКВА

Аннотация. Рассматривается особый случай включения в поэтический текст инородных дискурсивных элементов, который выражен как выстраивание стихотворения в форме исполняемой компьютерной программы. В поле исследования не попадают случаи репрезентации компьютерно порождаемой литературы, стихи о программистах и компьютерах. Речь идет о специальных (пока довольно редких случаях) соединения поэтической и программной формы. Предпосылкой такого симбиоза служит то, что исходный код программы и стихотворение являются текстом (последовательностью символов).

Предлагаются основания для деления поэтических текстов, выстроенных в форме программного кода. Это деление зависит от степени сложности избираемого языка, функциональной нагрузки написанной программы, а также от зависимости созданного текста от естественного языка.

[*] Исследование выполнено за счет гранта Российского научного фонда (проект № 14-28-00130) в Институте языкознания РАН.

Орехов Б. В. Стихи в программном коде: современный опыт и методика анализа // Критика и семиотика. 2015. № 1. С. 377–389.

ISSN 2307-1737. Критика и семиотика. 2015. № 1

Ключевые слова: поэтический дискурс, анализ поэтического текста, языки программирования.

УДК 82-193

Контактная информация: Орехов Борис Валерьевич, кандидат филологических наук, доцент Школы лингвистики факультета гуманитарных наук, НИУ «ВШЭ» (Старая Басманная ул., д. 21/4, Москва, 125009, borekhov@hse.ru)

В современные поэтические практики легко инкорпорируются элементы инородных поэзии дискурсов. Поэты активно используют графические и семантические маркеры техногенной перспективы (о некоторых примерах см.: [Суховей, 2003]).

В этой статье речь пойдет о ситуации пересечения стихотворного текста и программного кода, в которой поэтическое произведение одновременно является и исходными текстами компьютерных инструкций, предназначенных для исполнения электронным вычислительным устройством.

В некоторой мере такой симбиоз можно считать естественным. Поэзия изначально создается в текстовом выражении. Аналогичным образом и код современных компьютерных программ пишется в виде текста, т. е. понятной человеку последовательности символов, которые уже затем при помощи специальных алгоритмов транслируются в электрические сигналы и исполняются процессором компьютера.

Так, случайный пример компьютерной программы на одном из языков программирования выглядит следующим образом:

```
def checkValidKey(key):
    keyList = list(key)
    lettersList = list(LETTERS)
    keyList.sort()
    lettersList.sort()
```

В этом отрывке легко различимы не только отдельные символы, но и заимствованные из английского языка распространенные слова *check*, *valid*, *key*, *list* и др. Следовательно, некоторые предпосылки для монтирования поэтического дискурса и программного кода заложены в самом канале передачи информации. Их потенции не очевидны для непрофессионального взгляда, однако их реализацию мы вполне могли бы ожидать в современной поэтической практике.

Но некоторое место в начале статьи вынужденно будет посвящено более точному определению ее предмета. В него вовлечены особого рода

тексты, которые одновременно являются и исполняемыми программами, и стихами в авторском понимании. Иначе говоря, в объем нашего предмета не включаются написанные на естественном языке стихи о программистах, стихотворения, включающие какие-либо семантически маркированные символы ({ }, @, # и др.), а также «дигитальная литература», т. е. созданная при помощи технических средств. Последнее требует особого пояснения. В некотором смысле анализируемый здесь поэтический сегмент невозможен без существования компьютера, что является одним из дифференциальных признаков дигитальной литературы [1]. Действительно, чтобы программный код сохранял свою функциональность, он должен исполняться на компьютере. В то же время дигитальная литература обычно подразумевает интерактивное взаимодействие пользователя и программы, а также некоторый специфический компьютерный антураж, т. е. включенность компьютера как объекта в процесс поэтического перформанса. В случае с поэтическим произведением – компьютерным кодом все это оказывается вторичным. Анализируемые здесь тексты могут существовать независимо от электронных вычислительных устройств и даже могут быть опубликованы как исходные коды программ.

В качестве отрицательного примера можно привести программу на языке Perl, написанную в рамках эстетической стратегии дигитальной литературы:

```
perl -e '{print"a"x++$...$"x$.,$,=_;redo}' [2]
```

Эта программа, созданная Ником Монтфортом, имеет название «Все имена бога», и результатом ее работы оказывается автоматически порождаемая последовательность «имен»:

```
..._pygj_pygk_pygl_pygm_pygn_pygo_pygp_pygq_pygr_pygs_pygt_pygu_
pygv_pygw_pygx_pygy_pygz_pyha_pyhb_pyhc_pyhd_pyhe_pyhf_pyhg_pyhh
_pyhi_pyhj_pyhk_pyhl_pyhm_pyhn_pyho_pyhp_pyhq_pyhr_pyhs_pyht_pyhu_
pyhv_pyhw_pyhx_pyhy_pyhz_pyia_pyib_pyic_pyid_pyie_pyif_pyig_pyih_pyii
_pyij_pyik_pyil_pyim_pyin_pyio_pyip_pyiq_pyir_pyis_pyit_pyiu_pyiv_pyiw_
pyix_pyiy_pyiz_pyja_pyjb_pyjc_pyjd_pyje_pyjf_pyjg_pyjh_pyji_pyjj_pyjk_py
jl_pyjm_pyjn_pyjo_pyjp_pyjq_pyjr_pyjs_...
```

Очевидно, что поэтическая форма здесь связана с результатом работы программы (т. е. с порождением текста стихотворения) более, чем с самим исходным кодом.

---

[1] *Шмидт Э.* Буквальная (не)движимость. Дигитальная поэзия в РуЛиНете. URL: http://www.netslova.ru/schmidt/digital.html

[2] См.: http://nickm.com/poems/concrete_perl/

Положительный пример был опубликован на профессиональном ресурсе программистов «Хабрахабр» 9 июля 2013 г. пользователем YourDestiny:

```
if (newGame) resources.free();

s = FILENAME + 3;

setLocation(); load(s);

loadDialog.process();


try { setGamerColor(RED); }

catch(Exception e) { reset(); }

while (notReady) { object.make();

if (resourceNotFound) break; }


byte result; // сменить на int!

music();

System.out.print(""); [3]
```

Этот текст, как и другие исходные тексты программ на языке Java, можно запустить на компьютере, при этом акценты «поэтического» явным образом расставлены автором так, чтобы в фокусе была сама программа, а не результат ее работы. В данном случае чтение кода вслух возможно с выдерживанием строгого четырехстопного хорея.

Как уже было сказано, примеров текстов, подходящих под сформулированное определение, не так много, но вокруг них уже формируется оригинальная субкультура. В университете Стэнфорда к настоящему времени прошло три поэтических фестиваля для поэтов, пишущих стихи в виде компьютерного кода «Code Poetry Slam» (20 ноября 2013 г., 27 февраля

[3] http://habrahabr.ru/post/186044/

2014 г. и 23 января 2015 г. [4]). Девизом поэтических слэмов стал «create! compose! compile!» («создавай! сочиняй! компилируй!»).

К участию в программе мероприятия принимался код не только на языках программирования в строгом понимании этого термина, но и на языках разметки, т. е. в виде формализованного описания, служащего, например, для графического оформления интернет-страниц. Так, один из текстов второго слэма представлял собой код каскадных таблиц стилей:

Capsized Zak Kain, **Surrogate: Christina Hall**

```
.ocean {
   color: cornflowerblue;
   pitch: high;
   overflow: visible;
}

.boat {
   color: firebrick;
   transform: rotate(94deg);
   float: none;
}

.rescue-team {
   visibility: visible;
}

.crew {
   widows: none;
}
```

Этот код описывает внешний вид элементов интернет-страницы, которые могут быть названы *ocean*, *boat*, *rescue-team* и *crew*, т. е. эти имена задаются программистом. Однако автор оставляет читателю возможность воспринимать код как своеобразно организованный нарратив о перевернувшейся в океане лодке.

Еще одно структурное свойство, объединяющее программный код и поэтический текст, – явное формальное деление на строки-сегменты, отчетливо проступающее еще в одном примере с поэтического слэма в Стэнфорде:

[4] http://stanford.edu/~mkagen/codepoetryslam/

Apache Code Errors Aimee Norton

201 created
200 OK
100 continue
200 OK

303 see other
302 found
303 see other
409 conflict

403 forbidden
520 origin error
402 payment required
413 too large

303 see other
405 not allowed
417 expectation failed
423 locked down

502 bad gateway
307 redirect
204 no content
205 reset

305 use proxy
422 unprocessable entity
426 upgrade required
409 conflict

415 unsupported
429 too many requests
416 not satisfiable
417 failed

306 switched proxy
444 no response
449 retry
511 authenticate

301 moved permanently
401 unauthorized
506 variant negotiates
523 declined

406 not acceptable
451 illegal

599 timeout (unknown)
424 failed dependency

496 no certificate
423 locked away
598 timeout
598 timeout

В данном случае мы тоже видим своеобразный пример, это перечисленные в определенном порядке коды ошибок веб-сервера (т. е. специальной программы, отвечающей за взаимодействие физического сервера и пользовательской программы для навигации по Интернету) Apache. В таком виде эти строки, совершенно корректные технически, также создают лирический модус и ассоциируются с жизнью человека.

Разумеется, на странице поэтического слэма можно найти примеры, в большей степени отвечающие именно понятию «программный код», однако для понимания авторской интенции в этих случаях нужны специализированные программистские навыки:

for every i, **there is a stronger us Charles Mulloy**

```
#include

using namespace std;


class For_every
{
public:

   void ngular(){cout << endl << "I";};
   void que(){cout<<"f";};
   void ng(){cout<<" w";};
   void ghting(){cout << " d";};
   void vidual(){cout << "e";};
```

```
    void ving(){cout << "e";};
    void on(){cout << " n";};
};
class there_is_a_stronger
{
public:
    void t(){cout << "ot ";};
    void _(){ cout << "hang ";};
    void taining(){cout << endl << "together,";};
    void ting(){ cout << "we will ";};
    void tling(){cout<< "surely " << endl;};
    void ed(){cout << "seperately" << endl;};

};

int main()
{
        For_every i; there_is_a_stronger us;

                    /*I*/

        /*s*/i.ngular(); /*un*/i.que();

    /*try*/i.ng(); /*str*/i.ving(); /*f*/i.ghting();

/*ind*/i.vidual(); /*isolat*/i.on(); /*tr*/us.t(); /*foc*/us._();

    /*s*/us.taining(); /*adj*/us.ting(); /*b*/us.tling();

        /*victori*/us._(); /*f*/us.ed();

                /*Us*/

}
```

Здесь нехитрая с точки зрения интеллектуальной глубины идея соотношения понятий «я» и «мы» выражена с помощью имен переменных (использован язык C++), среди которых есть традиционно используемая для инкрементирования *i*.

Одним из основных исследовательских вопросов представляются методические основания, с которыми исследователь должен подходить к таким произведениям. Прежде всего, логичным было бы перечислить осно-

вания для деления текстов, это позволило бы установить общее и различное между ними.

Во-первых, важным критерием при анализе должен быть выбранный автором язык программирования. В практике программистов одни языки приобрели статус «простых», другие – «сложных», от этого качества зависит «скорость разработки», т. е. быстрота написания прикладных программ. С одной стороны, существуют скриптовые языки (Perl, Python, PHP), которые могут служить одновременно и языками для создания крупных промышленных проектов, и языками, на которых в силу их простоты удобно учиться программированию. Они называются языками высокого уровня, т. е. максимально абстрагированного от машинного кода и приближенного к человеку. С другой стороны, существуют языки низкого уровня, более ориентированные не на человека, а на «технический этаж» электроники (например, разные диалекты Ассемблера).

В данном случае выбор языка высокого или низкого уровня для написания текста может ассоциироваться с тем, ориентируется автор на широкую аудиторию (со всеми оговорками) или делает установку на элитарность.

Из общих соображений мы склонны считать, что автор будет сочинять текст на том языке, который он лучше всего знает и на котором чаще всего упражняется в своей профессиональной деятельности. Написание стихов становится для него своего рода терапевтической процедурой, снимающей эффект усталости от каждодневных занятий (здесь в силу вступает механизм карнавализации), а также демонстрацией своего профессионализма, виртуозности владения программистскими средствами. Наверное, не стоит исключать из этой схемы и интенцию показать творческое начало, важное для современной корпоративной культуры: не стоит забывать, что основными работодателями программистов являются средние и крупные компании.

Santa Claus Ying Hong Tham

```
import System.IO
import Data.List
import Data.Function
import qualified Data.Map as Map

type Karma = Bool -- True is nice, False is naughty

main = do
  contents <- readFile "childrenList.txt"
  let childrenKarma = map parseInputLine . filter isNotComment . lines $
contents
```

```
  let (niceList, naughtyList) = partition snd childrenKarma
  writeFile "naughtyList.txt" (unlines . map fst $ naughtyList)
  writeFile "niceList.txt" (unlines . map fst $ niceList)
  if checkLists naughtyList niceList -- checking list twice is redundant
    then putStrLn "Time to go to town."
    else putStrLn "Lists have errors. Repartition children."

checkLists :: [(String, Bool)] -> [(String, Bool)] -> Bool
checkLists naughtyList niceList =
  if all (not . snd) naughtyList && all snd niceList
    then True
    else False

parseInputLine :: String -> (String, Karma)
parseInputLine inputLine =
  (name, (read stringNum :: Double) >= 0) where
    [name, _, stringNum] = groupBy ((==) `on` (== ';')) inputLine
```

Однако даже с учетом такого личностного фактора языки, как уже было сказано, не равны между собой. При этом профессиональные программисты чаще всего умеют писать код сразу на нескольких распространенных языках, что в некотором смысле можно сопоставить с ситуацией ди- или полиглоссии, характерной для высокого Средневековья, когда, к примеру, Данте нужно было выбирать, на латыни или на итальянском языке создавать свою главную поэму.

Кроме сложности, у части языков есть свой круг традиционных задач. Python или C относятся к языкам общего назначения, а вот список ошибок веб-сервера Apache или каскадные таблицы стилей нужны в совершенно определенных (описанных выше) целях. Очевидно, эти цели также влияют на восприятие поэтического текста.

Кроме того, существуют устаревшие языки, вышедшие из употребления и не используемые (Cobol, Pascal, Algol). Написание кода на них также имеет аналогию в создании текстов на мертвых естественных языках.

Во-вторых, аналитически подходящий к таким текстам исследователь должен учитывать функциональную нагрузку программы. Что она призвана делать? Каждая компьютерная программа создается ради достижения некоторого результата. Ясно, что в случае с поэтическим программным кодом этот результат может не быть прагматически ориентированным, но он в любом случае должен учитываться.

Кроме того, программисты уделяют большое внимание тому, насколько оптимален способ достижения результата. Одним из важных постулатов для программистов на Python является соображение, что есть только один (правильный) способ сделать что-то, что нужно сделать. В то же вре-

мя важной в программистской практике оказывается и гибкость языка, отразившаяся в афоризме создателей языка Perl: «Существует много способов сделать это». Очевидно, что в поэтической практике гибкость языка будет цениться больше, однако и бедный язык будет налагать компенсирующие ограничения, которые могли бы заставить автора проявить изобретательность.

Вот текст на языке C, формализованным образом описывающий зависимость или увлеченность. Его программистская особенность в создании бесконечного цикла, который чаще всего считается не лучшим способом реализовать прикладную задачу:

addiction Oishi Banerjee

```
void addiction(){
    int mind = 1;
    while (true){
        //what goes around wraps around
        //and we climb the peaks to find they touch the abyss
        mind = mind+1;
        if (mind<0){
            break;
        }
    }
}
```

В-третьих, исследователь обязан обратить внимание на наличие зависимости анализируемого текста от естественного языка. Может ли программа существовать без родного языка программиста (например, английского)? Хватает ли выразительных средств языка программирования? Может ли автор сказать все, что нужно, используя только выразительные средства языка программирования? Нужно особо остановиться на том, что поэт, сочиняющий такого рода текст, оказывается в ситуации поиска новых выразительных форм. Его стремление состоит в том, чтобы отказаться от естественного языка (английского, русского, французского). И действительно, если ему удастся создать такое произведение, которое было бы понятно человеку, знакомому с языком программирования, на котором оно написано, но не знакомому с родным языком программиста, автор на этом пути смог бы достичь успеха и тем самым фактически создать новую художественную форму, а также вовлекать в круг своей аудитории не зависящих от языка реципиентов. Так это происходит с живописью и скульптурой, в которых круг воспринимающих зависит от погруженности в художественный язык, но не естественный язык.

Такого рода эксперименты редки. Вот обычный случай:

INSERT `your hand` INTO `mine` SELECT `restaurant` FROM `nearby` WHERE `you would` LIKE 'to date';

Здесь используются операторы языка SQL, но фактически это текст на английском языке.

Возможность вводить сколь угодно новые для языка слова, которые ничего не значат с точки зрения дизайна языка, но знакомы воспринимающему благодаря их аналогам в естественном языке, слишком соблазнительна и постоянно используется авторами. Думается, это своего рода облегчение задачи, аналог грамматической рифмы в традиционной поэзии.

Сформулированный исследовательский вопрос напрямую смыкается и с тем, предназначены ли стихи для того, чтобы их декламировали (как первый приведенный пример с сервиса «Хабрахабр»)? Альтернативой декламации для текстов подобного рода должно быть воспроизведение на экране. Отметим, что декламация слишком привязана к традиционным поэтическим формам.

Главный принцип рассматриваемых текстов в соединении разнородного материала. Когда прием контрастности будет исчерпан, практика создания таких стихотворений должна будет сойти на нет. Трудно предсказывать наступление этого момента, так как за этой практикой стоят, кроме собственно художественной интенции, также и терапевтические механизмы. Однако общий вывод должен состоять в том, что со временем в руках исследователей окажется законченный корпус поэтических программ.

Список литературы

Суховей Д. Круги компьютерного рая. Семантика графических приемов в текстах поэтического поколения 1990–2000 годов // НЛО. 2003. № 62. С. 212–241.

Article metadata

Title: Verse as a program code: contemporary experience and analysis methods

Author: B. V. Orekhov

Author's e-mail: borekhov@hse.ru

Author affiliation: Higher School of Economics

Abstract: The article deals with the special case of inclusion of a foreign discursive elements in the poetic text. This case articulated as a poems written in the form of an executable computer program. This study field does not in-

clude cases of computer generated literature, poems that takes programmers and computers as a main subject. This is a special (we still have very rare cases) connections between poetic and software form. The base of this symbiosis is that the source code of the program and the poem is text (sequence of characters).

The article offers a framework for the division of poetic texts of this kind. This division depends on the degree of complexity of the elected programming language, on the functionality of the program, on the dependence of the generated text from a natural language.

Key terms: poetry discourse, poetic text analysis, programming languages.

Reference literature (*in transliteration*)*:*

Suhovej D. Krugi komp'juternogo raja. Semantika graficheskih priemov v tekstah pojeticheskogo pokolenija 1990–2000 godov // NLO. 2003. № 62. S. 212–241.