**УДК 004.4'22**

**А.П. Соколов**1,**\***, канд. физ.-мат. наук, доц., E-mail: [alsokolo@bmstu.ru](mailto:alsokolo@bmstu.ru)

**А.Ю. Першин**1, ассистент, E-mail: [tony.pershin@gmail.com](mailto:tony.pershin@gmail.com)

1Московский Государственный Технический Университет имени Н.Э. Баумана, Москва, Россия

**Система автоматизированного проектирования**

**композиционных материалов**

*Современный процесс проектирования новых материалов невозможен без применения специализированных программных систем.*

***Ключевые слова:*** ; технологии разработки систем инженерного анализа.

**Введение**

Процесс ведения разработки современного прикладного программного обеспечения (ПО) предполагает, помимо прочего, необходимость написания существенного объема исходных кодов программ, что может быть реализовано только при слаженной работе команды разработчиков. При разработке больших программных систем всегда встает задача согласованного повторного использования исходного кода. При любом дублировании исходного кода могут возникать копии ошибок, а с увеличением масштаба разрабатываемого ПО количество таких ошибок возрастает многократно. Сокращение числа ошибок в конечном ПО обычно обеспечивается следующими способами:

* регламентацией и стандартизацией процессов разработки;
* созданием таких условий для разработки, при которых с увеличением числа функциональных возможностей системы сохраняется линейный рост сложности внесения изменений в каждую отдельную функцию (реализуется глубоко продуманной архитектурой разрабатываемого ПО, реализацией возможностей гибкого расширения функциональных возможностей и пр.);
* применением специализированных программных средств автоматизированной поддержки процессов разработки (CASE - инструментариев, англ. Computer-aided Software Engineering Tools);

Применение средств автоматизации процессов разработки ПО является признаком высокого уровня зрелости команды разработки, и, в особенности, оно необходимо при создании крупных расширяемых программных комплексов [1]. Среди прочих[[1]](#footnote-2) в состав CASE - инструментариев входят программные средства генерации кода [2] различных типов.

Применение методов автоматизированной генерации исходного кода программных модулей и документации началось еще в 50-х годах прошлого века при создании первых компиляторов [3-4]. Известны следующие подходы к генерации кода: а) преобразование графического представления программного объекта в исходный код (англ. model-to-text transformations (M2T)) [5]; б) генерация кода на основе шаблонов (англ. template-based code generation (TBCG)) [4].

Использование первого подхода сопряжено с применением парадигмы «модель-ориентированной» разработки программного обеспечения (англ. Model-driven engineering (MDE)) [5]. В англоязычных публикациях в контексте MDE используют термин «модель объекта», под которым понимают структуру классов, реляционную модель данных и пр. В рамках MDE с использованием некоторого языка моделирования (например, UML, IDEF0, ARIS [5, 6]) осуществляется построение «модели объекта», ее интерпретация и построение заготовок исходного кода программного обеспечения. Примеры прикладного применения генераторов кода на основе графического представления элементов программных систем (иерархий классов, реляционных моделей данных и пр.) представлены во многих работах, например [7, 8]. В настоящей работе эти подходы, несмотря на их широкое использование, не рассматриваются.

Предметом настоящей статьи являются методы генерация кода на основе шаблонов, обеспечивающие автоматизированное формирование некоторого текстового результата (объект генерации), соответствующего заранее определённому формату, называемому шаблоном (рисунок 1). Программные средства, реализующие соответствующие методы, называют «генераторами кода на основе шаблонов».



Рисунок 1 – Традиционная схема процесса генерации кода на основе шаблонов

Применение генераторов кода на основе шаблонов обеспечивает прототипирование программного обеспечения, а именно - автоматизирует процедуры создания заготовок (прототипов): исходного кода отдельных программных структур данных; программных модулей на различных языках программирования (например, высокоуровневых языках C++, Java и др.); исходного кода программной документации, разрабатываемой, например, с использованием языка LaTeX; сценариев на языках систем автоматизации сборки (например, CMake) и т.д. Очевидно, такие возможности могут существенно ускорить процессы разработки программного обеспечения.

Процедуры генерации принято различать в зависимости от типа применяемых шаблонов [4]: а) на основе предварительно определенных шаблонов (англ. Predefined) или «зашитых» в генерирующую программу; б) основанные на формате генерируемого выходного объекта (англ. Output-based); в) основанные на множестве формализованных правил (англ. Rule-based).

Традиционно шаблоны состоят из двух частей: статической (постоянной при генерации) и динамической (изменяемой при генерации). Предварительно определенные шаблоны (тип 1) представляют собой такой класс шаблонов, для которых статическая часть может модифицироваться, а результат динамической сильно зависит от применяемого генератора. Шаблоны, основанные на формате генерируемого выходного объекта (тип 2), представляют собой такие текстовые данные, которые синтаксически соответствуют требуемому на выходе результату генерации. Такой класс шаблонов наиболее часто применяется на практике. Наконец, шаблоны, основанные на правилах (тип 3), предполагают возможности определения формальных правил генерирования непосредственно в динамической части тела шаблона, указывая на необходимость применения специальных алгоритмов вычисления значений выходных параметров. Генераторы, поддерживающие такие шаблоны, обычно более сложные, так как должны обеспечивать интерпретацию соответствующих правил, включенных в состав шаблона. Правила в свою очередь могут быть определены независимо.

Среди известных программных средств и технологий, применяемых для генерации кода на основе шаблонов различных типов следует отметить (в порядке снижения популярности согласно [4]): Acceleo, Xpand, XSLT, Velocity, JET, Fujaba, Simulink TLC, EGL, Rational, String Template, MOF Script, Free Marker, Rhapsody, Xtend и многие другие.

Отметим, что большинство из представленных программных средств для их использования предполагают оплату лицензии [9], а существующие бесплатные разработки обладают либо ограниченными функциональными возможностями, либо они узкоспециализированы. В обзоре [4] и в докладе [10] представлено подробное сравнение известных систем генерации кода на основе шаблонов.

Генераторы кода, в том числе на основе шаблонов, широко применяются при создании ПО различного назначения: системы, использующие базы данных [11, 12], встраиваемые системы реального времени [13-15], web-ориентированные системы [16], инженерные программные системы [17] и прочие. Отдельный интерес представляют генераторы кода, использующие шаблоны, разрабатываемые на специализированном языке [18, 19], либо формирующие код на основе предварительного определения многоуровневого набора правил и без формирования шаблонов вовсе [20].

Информации об отечественных программных средствах генерации кода, доступной в открытой печати, достаточно мало. Например, в работе [21] кратко представлена система автоматизации процесса разработки АТ Studio 2015, которая разрабатывается группой компаний «ИВС».

Многообразие существующих решений, в том числе и вновь разрабатываемых [11, 12, 14-18, 20, 21], по мнению авторов, связано с принципиальной важностью развития собственных CASE инструментариев при создании коммерческого программного обеспечения. Использование в коммерческих разработках сторонних CASE инструментариев приводит к серьезной внешней зависимости организации от последних.

Несмотря на длительную историю, в области генерации кода до сих пор существуют требующие решения актуальные задачи [4], связанные, например:

* со снижением трудоёмкости процессов разработки и обновления шаблонов с учётом изменяющихся требований к конечным объектам генерации[[2]](#footnote-3);
* с применением методов генерации кода при разработке наукоёмкого программного обеспечения, в частности, при создании систем математического моделирования и инженерного анализа[[3]](#footnote-4);
* с анализом эффективности процессов генерации кода, в том числе, на основе шаблонов, при создании программного обеспечения различного назначения и масштаба.

В рамках проводимых научно-исследовательских и опытно-конструкторских работ авторами ведутся разработки прикладного программного обеспечения инженерного анализа. В частности, идет разработка расширяемого программного комплекса «Распределенная вычислительная система GCD» (РВС GCD) [22]. С **целью** повышения эффективности и качества процесса разработки в целом, уменьшения количества ошибок при создании новых подсистем и программных модулей, автоматизации процессов разработки документации, а также в целях обеспечения независимости ядра и других компонентов системы от стороннего программного обеспечения, был создан представляемый в статье программный инструментарий [23, 24].

Представленные в статье решения основаны на принципах и подходах, активно применяемых при разработке и внедрении промышленных информационных систем. Созданный и представленный в статье программный инструментарий стал основой для разработки CASE инструментария РВС GCD [24].

**Актуальность** разработки обусловлена существенной трудоёмкостью, а часто и невозможностью, создания расширяемого и сопровождаемого программного обеспечения без применения вспомогательных автоматизирующих процессы разработки программных средств.

**Целью** публикации является представить отмеченный выше программный инструментарий, позволяющий автоматизировать процессы разработки программного обеспечения и документации, включая каркасы программных компонент, проекты библиотек, модулей расширения, файлов исходных данных для произвольных функций системы и прочие объекты. Ключевой составляющей инструментария стала библиотека функций генерации кода на основе шаблонов.

1. **Проектирование программного инструментария**

К разработанному программному инструментарию предъявлялись следующие функциональные требования.

1. Формат шаблона должен быть текстовым.
2. Должна быть реализована поддержка шаблонов, зависящих от других шаблонов.
3. Должны быть реализованы функции регистрации шаблонов и их атрибутов, включая расположение шаблона, в специальной базе данных.
4. Функции генерации не должны зависеть от выбранного шаблона и его параметров.
5. Должна быть реализована поддержка параметров различных типов: а) скалярные и векторные параметры; б) статические (терминальные и нетерминальные) и динамические (функциональные) параметры; в) общие (шаблон-независимые) и дополнительные (шаблон-зависимые) параметры.
6. Список параметров определенного шаблона не должен быть фиксирован, должна оставаться возможность его расширения.
7. Должна быть обеспечена поддержка возможностей загрузки значений параметров из различный источников: файла входных данных; файла дополнительных параметров, специфичных выбранному шаблону; базы данных; объекта специального класса (с целью возможностей интеграции с другими системами).
8. Должна быть реализована возможность преобразования получаемых из внешнего источника данных в произвольный текстовый вид в зависимости от синтаксиса определения параметра в шаблоне.
9. В зависимости от типов используемых в шаблоне параметров должны применяться различные алгоритмы вычисления значений параметров.
10. Подготовка списка дополнительных параметров и их значений при создании нового шаблона или редактировании существующего должна быть доступной для неподготовленного специалиста, не владеющего навыками программирования.
11. Непосредственный ввод значений параметров объекта при генерации должен быть обеспечен с и без использования графического пользовательского интерфейса (GUI).
12. Процесс построения GUI должен осуществляться во время выполнения программы при запросе значений параметров для выбранного пользователем шаблона до генерации.
13. Программный инструментарий должен обладать возможностью встраивания в web-ориентированные приложения, приложения для мобильных платформ и приложения для операционных систем семейств Windows, Linux.

В следующем разделе представлены определения понятий, которые стали основой для разработки архитектуры программного инструментария генерации кода на основе шаблонов. Далее, для краткости, будем использовать термины: «генерация на основе шаблонов» или, просто, «генерация».

* 1. **Базовые понятия**

В Таблице 1 представлен перечень специальных символов и обозначений. При конкретной программной реализации указанные символы могут быть заменены на иные с сохранением семантики.

Далее представлены введённые вспомогательные определения.

*Определение 1. Скалярным параметром* шаблона будем называть строковую константу **@**ParamName**@**, слева и справа ограниченную символом ‘**@**’.

Имя параметра ParamName рекомендуется определять с использованием нотации CamelCase (таблица 1).

Таблица 1 – Применяемые специальные обозначения

|  |  |  |
| --- | --- | --- |
| **Обозначение** |  | **Описание** |
| ‘@’ | - | символ используется для отделения скалярного параметра от остального текста в составе определения шаблона; |
| «скобки» ‘%#...#%’ | - | используются для отделения векторного параметра от остального текста в составе определения шаблона; |
| ~ | - | символ используется в контексте A ~ B, что будет обозначать, что B есть тип объекта A; |
| char(N) | - | обозначение строкового типа, объекты которого должны состоять ровно из N символов; |
| varchar(N) | - | обозначение строкового типа, объекты которого должны состоять не более чем из N символов; |
| […] | - | квадратные скобки используются для определения фильтров при вычислении значений векторных параметров; |
| {…} | - | фигурные скобки используются для обозначения необязательных данных, при этом при подстановке значений эти скобки следует удалить; |
| lc | - | указание на использование нижнего регистра (lower case); |
| uc | - | указание на использование верхнего регистра (UPPER CASE); |
| CamelCase | - | указание на использование нотации CamelCase (в одно слово, английскими буквами, каждое слово с большой буквы, остальные строчные). |

*Определение 2. Векторным параметром* шаблона будем называть текстовое выражение, соответствующее представленному далее синтаксису.

|  |
| --- |
| **%#@**DataSource**@**{<filter>}**:@**OutputFormat**@#%**  **@**DataSource**@** = {<db\_table\_name>|<xdbt\_table\_name>}  <filter> = **[**<attr\_i\_name>=<value> [{AND|OR} ...]**]**  <value> = {<string\_value>|@ScalarParamName@|@DBTablePKValue@.<attr\_j\_name>}  **@**OutputFormat**@** = <name\_of\_output\_format> |

В этом выражении: @DataSource@ - скалярный параметр, значение которого должно быть либо именем таблицы базы данных, либо именем файла в формате XDBT (XML формат для описания таблицы базы данных и ее контента после выгрузки, применяемый в РВС GCD);

<filter> - необязательный параметр, определяющий одно или несколько условий отбора данных из указанного источника **@**DataSource**@**, где <attr\_i\_name> - имя некоторого i-го атрибута источника;

<value> - значение атрибута, задаваемое либо явно в виде строковой константы <string\_value>, либо неявно ссылкой на некоторый скалярный параметр @ScalarParamName@, либо неявно ссылкой на скалярный параметр @DBTablePKValue@.<attr\_j\_name>, вычисляемый автоматически в предположении, что @DBTablePKValue@ - имя параметра, на основе которого могут быть определены: идентификатор некоторого другого источника векторных данных (таблицы базы данных) и значение атрибутов, идентифицирующих единственный объект этого источника, тогда как <attr\_j\_name> - определяет имя атрибута источника, определяемого на основе @DBTablePKValue@;

<db\_table\_name> - имя таблицы базы данных, доступ к которой предполагается обеспеченным;

<xdbt\_table\_name> - имя файла в формате XDBT, представляющего либо таблицу базы данных, либо результат запроса к базе данных в виде предварительно выгруженного XML файла, доступ к которому предполагается обеспеченным;

<name\_of\_output\_format> - имя формата представления результата, зарегистрированного в базе данных системы, которому ставится в соответствие конкретный метод преобразования, основанный, например, на XSL - преобразовании.

*Определение 3.* *Шаблоном* генератора кода будем называть непустой каталог в файловой системе, который может включать в свой состав как файлы, так и каталоги с файлами, обладающий следующими свойствами: имена файлов и каталогов могут включать определения скалярных параметров; содержание текстовых файлов, может включать определение как скалярных так и векторных параметров.

*Определение 4.* Шаблон A будем называтьзависящим от шаблона B, если среди файлов, определяющих шаблон A, присутствует файл с именем B.ref.

Скалярные параметры шаблонов подразделяют на статические и динамические в зависимости от метода вычисления значения параметра.

*Определение 5.* Скалярный параметр будем называть *динамическим*, если его значение может быть вычислено исключительно путём запуска специальной программной функции.

*Определение 6.* Скалярный параметр будем называть *статическим*, если его значение задано явно в исходных данных (файлах исходных данных или специальных таблицах базы данных) и/или может зависеть от значений других скалярных параметров статических и/или динамических.

Приведем пример значений статических параметров.

|  |
| --- |
| @Param1@ = value1  @Param2@ = @Param1@value2  @Param3@ = value3@Param2@value2  @Param4@ = @DynamicParam@ |

В представленном примере: @Param1@ - скалярный статический терминальный параметр с постоянным явно заданным значением;

@Param2@ и @Param3@ - скалярные статические нетерминальные параметры с постоянными явно заданными значениями;

@DynamicParam@ - скалярный динамический параметр, значение которого вычисляется в зависимости от реализации функций генерации;

@Param4@ - скалярный статический параметр с неизвестным значением, определяемым значением скалярного динамического параметра @DynamicParam@.

При разработке шаблона используемые параметры следует разделять на шаблон-независимые и шаблон-зависимые.

*Определение 7.* Параметр будем называть *шаблон-независимым*, если он может использоваться в более, чем одном шаблоне.

*Определение 8.* Параметр будем называть *шаблон-зависимым*, если его использование возможно только в одном шаблоне.

*Определение 9.* Шаблон будем называть *гибридным*, если в составе его признаков можно выделить признаки шаблонов на основе форматов выходных данных и шаблонов, основанных на правилах.

В контексте представляемой разработки все используемые шаблоны гибридные, поэтому здесь и далее под шаблонами подразумеваются гибридные шаблоны.

* 1. **Использование параметров шаблонов векторного типа**

При использовании параметров векторного типа источник данных, как и формат представления результата, могут быть указаны в шаблоне явно, однако, это формирует зависимость шаблона от конкретных источника данных и/или формата представления результата соответственно.

Для устранения указанных зависимостей рекомендуется использовать скалярные параметры шаблонов в составе векторных параметров. Представим сказанное на примере (листинг 1).

Листинг 1. Фрагмент кода некоторого шаблона с неявным определением векторного параметра, зависящего от значений скалярных параметров: @DataSource@ – источник данных; attr1 – идентификатор существующего атрибута источника данных @DataSource@; @value@ – требуемое значение атрибута; @OutputFormat@ – идентификатор выходного формата

|  |
| --- |
| …  %#@DataSource@[attr1 = @value@]:@OutputFormat@#%  … |

* 1. **Особенности фильтрации значений параметров шаблонов векторного типа**

В дополнение к представленному выше примеру (листинг 1) следует заметить, что значение @value@ должно быть каким-то образом определено, а именно: введено разработчиком шаблона явно; задано статически среди входных параметров шаблона (например, в коде).

В рамках РВС GCD для определения значений входных параметров применяется текстовый формат aINI (листинг 2). Подробное описание в работе [25].

Листинг 2. Синтаксис языка aINI в расширенной форме Бекуса-Наура с комментариями, где для краткости опущены определения нетерминальных символов ID (строковый идентификатор) и TEXT (строковая константа)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | **aini\_file** = [sections]. (\*aINI файл состоит из секций\*)  **sections** = section ["\n" sections]. (\*Секций может быть произвольное количество\*)  **section** = [comment "\n"] "[" ID "]" [comment] "\n" parameters. (\*До и после секции возможны комментарии. Секции состоят из параметров\*)  **comment** = "//" TEXT.  **parameters** = [ [("-"|"\*")] parameter "\n"] [parameters].  **parameter** = (sub\_parameter\_id | ID) "=" value ["//" comment].  **sub\_parameter\_id** = set\_id "$" ID. (\*Идентификатор субпараметра\*)  **value** = (text\_value | bool\_value | file\_ref | dbtable\_ref | dim\_value | array\_value | interval\_value | combobox\_value).  **text\_value** = TEXT. (\*Текстовое значение\*)  **bool\_value** = "[" bool\_val "](0|1)". (\*Логическое значение\*)  **bool\_val** = ("0" | "1").  **file\_ref**="[" file\_name "]". (\*Ссылка на файл(возможно известного формата)\*)  **file\_name** = ID "." ext. (\*Имя файла\*)  **ext** = ID. (\*Расширение файла (определяет формат)\*)  **dbtable\_ref**="[" rec\_id "]$" schema "." tablename. (\*Ссылка на запись в таблице БД\*)  **schema** = ID. (\*Схема данных\*)  **tablename** = ID. (\*Имя таблицы\*)  **rec\_id** = pk\_f [";" rec\_id]. (\*Значения атрибутов первичного ключа таблицы, определяющих одну запись\*)  **dim\_value** = ID "[[" ID "]]". (\*Значение с размерностью\*)  **array\_value** = "(" array ")". (\*Значение массива\*)  **array** = single\_value [";" array].  **single\_value** = (TEXT | set\_id).  **set\_id** = ID.  **interval\_value** = "[" current ";" min "=" max ";" step "]". (\*Значение, задаваемое диапазоном\*)  **current** = TEXT. (\*Текущее значение\*)  **min** = TEXT. (\*Минимальное значение\*)  **max** = TEXT. (\*Максимальное значение\*)  **step** = TEXT. (\*Предпочтительный шаг\*)  **combobox\_value** = "[" current\_id "](" ID [";" ID] ")". (\*Множество с текущим выбранным элементом\*)  **current\_id** = ID. (\*Текущий выбранный идентификатор\*) |

Пусть требуется вывести массив данных из одного источника (источник @DataSource1@), фильтруемых при известных значениях некоторых атрибутов другого источника (источник @DataSource2@), связанного с первым, например, внешним ключом. В таком случае указанный в листинге 1 синтаксис может быть обобщен (листинг 3).

Листинг 3. Фрагмент кода шаблона с неявным определением векторного параметра, зависящего от значений скалярных параметров: @DataSource1@, @DataSource2@ - источники данных

|  |
| --- |
| …  %#@DataSource1@[attr1 = @DataSource2.attr1@]:@OutputFormat@#%  … |

В случае, если определение значений параметров @DataSource1@ и @DataSource2@ осуществляется с использованием файла исходных данных и/или файла дополнительных параметров в формате aINI (листинг 2), то возможно существенно упростить процедуру определения значения @DataSource2.attr1@ за счет применения параметров типа «ссылка на запись в таблице базы данных» (листинг 2, строка 15).

Пусть определён aINI-файл исходных данных (листинг 4) постановки задачи генерации.

Листинг 4. Фрагмент aINI-файла исходных данных генератора кода

|  |
| --- |
| …  Template = [PAT]$sys.tmpls // Идентификатор шаблона  … |

Пусть определён aINI-файл дополнительных параметров (листинг 5), определяющий значения параметров, специфичных выбранному шаблону.

Листинг 5. Фрагмент aINI-файла дополнительных параметров шаблона

|  |
| --- |
| …  DataSource1 = sld.matpr // Параметры материалов  DataSource2 = [03Х16Н15М3]$sld.matrl // Идентификатор материала  OutputFormat = [TXT]$gen.frmts // Выходной формат  … |

Параметр @DataSource2@ (листинг 5) имеет aINI-тип «ссылка на запись в таблице базы данных» [25], значением которого является идентификатор материала 03Х16Н15М3, определяющий значения атрибута(ов) первичного ключа источника данных sld.matrl.

В связи с тем, что значение первичного ключа определено, то, формально, при запросе данных из источника sld.matrl, соответствующих этому значению ключа, будет возвращена единственная запись, в составе которой могут присутствовать также и значения неключевых атрибутов. В таком случае, очевидной является возможность динамического дополнения списка скалярных параметров генератора во время выполнения новыми параметрами. Названия таких параметров должны соответствовать именам неключевых атрибутов, а значения – значениям этих атрибутов.

Для реализации возможностей последующего уникального выбора значений параметров из конечного списка, имена таких параметров при их дополнении должны быть уникальными. Обеспечение подобной уникальности реализуется алгоритмом формирования имен автоматически добавляемых параметров, имеющих очевидный синтаксис: @db\_record\_ref@.[attr], где @db\_record\_ref@ - имя параметра, имеющего aINI-тип «ссылка на запись в таблице базы данных» и присутствующего среди входных параметров задания на генерацию, [attr] - идентификаторы атрибутов анализируемой таблицы, загружаемые при запросе данных.

Например, пусть таблицы sld.matrl и sld.matpr определены согласно Листингам 6, 7. Ключевые атрибуты подчеркнуты.

Листинг 6. Пример определения таблицы sld.matrl (DataSource2)

|  |
| --- |
| CREATE TABLE sld.matrl  (  **matna** sld.matna NOT NULL DEFAULT sys.get\_next\_user\_sid('MAT'),  **dscra** com.dscra,  **rsobj** srg.rsobj,  CONSTRAINT matrl\_pkey PRIMARY KEY (**matna**),  CONSTRAINT matrl\_rsobj\_fkey FOREIGN KEY (rsobj)  REFERENCES srg.rsobs (rsobj)  ) |

Листинг 7. Пример определения таблицы sld.matpr (DataSource1)

|  |
| --- |
| CREATE TABLE sld.matpr  (  **matna** sld.matna NOT NULL,  **ioprm** com.ioprm NOT NULL,  **detmt** com.detmt NOT NULL,  **unitt** com.unitt NOT NULL,  sllnk com.dscra,  CONSTRAINT matpr\_pkey PRIMARY KEY (**matna, ioprm, detmt, unitt**),  CONSTRAINT fk\_sld\_matpr\_matna FOREIGN KEY (matna)  REFERENCES sld.matrl (matna)  ) |

Пусть в выбранном шаблоне PAT встречается синтаксическая конструкция согласно листингу 8.

Листинг 8. Фрагмент кода шаблона PAT.

|  |
| --- |
| …  В работе проводилось исследование эффективных термоупругопрочностных характеристик (ЭТУПХ) композиционного материала @DataSource2.matna@ (@DataSource2.dscra@).  Значения полученных ЭТУПХ представлены в таблице ниже.  %#@DataSource1@[matna = @DataSource2.matna@]:@OutputFormat@#%  … |

Таким образом, при использовании шаблона PAT список параметров генератора кода должен быть динамически расширен новыми параметрами согласно листингу 9.

Листинг 9. Дополнительные параметры, формируемые автоматически, включая определение их значений.

|  |
| --- |
| …  DataSource2.matna = 03Х16Н15М3  DataSource2.dscra = <Описание материала 03Х16Н15М3>  … |

Аналогичное дополнение новыми параметрами обеспечивается для любого параметра, имеющего aINI-тип «ссылка на запись в таблице базы данных». Операция дополнения осуществляться при формировании объекта на основе выбранного шаблона.

* 1. **Архитектура программного инструментария**

Первая версия инструментария была разработана Соколовым А.П. в 2007 году при создании программного комплекса GCAD v.2., предназначенного для автоматизированного прогнозирования свойств композиционных материалов [26]. В рамках разработки РВС GCD (GCAD v.3) [22] Соколовым А.П., Першиным А.Ю. и Макаренковым В.М. была создана вторая версия инструментария, архитектура которой представляется в настоящей статье.

Схема разработанного программного инструментария генерации кода в целом соответствует традиционной (рисунок 1). Однако, она имеет и некоторые особенности, которые отображены на рисунке 2. В частности, в состав инструментария вошла база данных, которая позволила сохранять: данные о доступных шаблонах и их местоположениях; настройки применяемых синтаксических анализаторов; данные об общедоступных параметрах, их типах и функциях их вычисления;  информацию о поддерживаемых преобразователях и их типах.



Рисунок 2 – Диаграмма потоков данных процесса генерации кода на основе шаблонов с использованием разработанного программного инструментария

* + 1. **Особенности программной реализации алгоритма генерации**

В процессе программной реализации алгоритма генерации кода третьей версии инструментария применялся графоориентированный подход [27]. В рамках данного подхода предполагается, что алгоритм программы представляется в форме ориентированного графа, для которого узлы являются состояниями данных, а с рёбрами связаны функции перехода, преобразующие данные из одного состояния в другое. Тройка объектов: ориентированный граф, множество состояний данных и множество функций перехода (пара функция-обработчик и функция-предикат) позволяют ввести понятие графовая модель метода обработки данных. Все указанные понятия формально введены в работе [28], детальное описание выходит за рамки настоящей статьи.

На рисунке 3 представлена графовая модель верхнего уровня абстракции, представляющая алгоритм работы генератора, где использованы обозначения для функций-обработчиков *fij* (Таблица 2), функций-предикатов *pij*, состояний данных *Sk*, а также not(*pij*) – обозначение оператора отрицания.



Рисунок 3 – Схема процесса генерации кода на основе шаблонов в рамках РВС GCD с представлением алгоритма генерации в виде его графовой модели

Таблица 2. Описания функций-обработчиков графовой модели алгоритма генерации кода на основе шаблонов

|  |  |  |
| --- | --- | --- |
| **#** | **Функция-обработчик** | **Комментарий** |
| *f12* | Подготовка массива подстановок, на основе скалярных параметров. | Источники параметров: а) файлы исходных данных и дополнительных параметры; б) база данных подсистемы генерации; в) неявно определяемые параметры на основе параметров aINI-типа «ссылка на запись в таблице базы данных» (см. раздел 1.3). |
| *f23* | Подготовка преобразуемого контента. | Копирование шаблона во временное местоположение. |
| *f33* | Замена имен файлов и каталогов. | Рекурсивная замена, во временном местоположении объекта. |
| *f44* | Преобразование контента на основе скалярных параметров. | Рекурсивное преобразование всех скалярных параметров, во временном местоположении объекта. |
| *f55* | Анализ и преобразование контента на основе векторных параметров. | Выявление запросов значений векторных параметров, вычисление значений и преобразование векторных параметров с использованием зарегистрированных в системе XSL-преобразователей. |
| *f56* | Подготовка преобразованного контента. | Перенос объекта в конечного местоположение или архивирование и отправка. |
| *f34, f45* | Вспомогательные функции. | - |

Применение графоориентированного подхода позволяет гибко заменять и/или обновлять любые функции-обработчики, а также естественным образом формировать библиотеки функций программного инструментария в целом.

* + 1. **Формат файла исходных данных задания на генерацию**

Для определения файла исходных данных задания на генерацию также использовался формат aINI (листинг 10).

Листинг 10. Шаблон GEN задания для генерации кода в формате aINI

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | [Author]//Идентификация автора разработки  **AuthorName**=[AuthorName]$sys.users//Автор разработки  **-AuthorSID**=@AuthorName.useri@//SID Автора  [Generator parameters]//Параметры генерации  \***CopyObjectToRep**=[0]{0|1}//Перенести объект генерации в репозиторий  \***RepPath**=@RepPath@//Целевой репозитории (URL)  \***TemplatesPath**=@TemplatesPath@//Репозиторий шаблонов (URL)  \***TemporaryPath**=@TemporaryPath@//Каталог временного хранения (URL)  [Object parameters]//Параметры генерируемого объекта  \***CodeObjectName**=@CodeObjectName@//Наименование объекта(varchar(25))  \***Description**=@Description@//Описание  **ParametersFile**=[ParametersFile.imp]//Дополнительные параметры  \***TemplateSID**=[@TemplateSID@]$gen.tmpls//Идентификатор шаблона  [Project data]//Идентификация объекта  \***ComplexSID**=[ComplexSID]$sys.cmplx//Идентификатор комплекса  \***SolutionSID**=[SolutionSID]$sys.solun//Идентификатор решения  \***ProjectSID**=[ProjectSID]$sys.prjct//Идентификатор проекта |

Указанные параметры являются статическими и задаваемыми для каждой новой процедуры генерации. Параметры, отмеченные звёздочкой, обязательны для заполнения, а отмеченные символом минус - скрытые.

Параметры CopyObjectToRep иTemporaryPath используются для указания необходимости размещения результата генерации во временном расположении, что зачастую обусловлено необходимостью отладки результата генерации и шаблона.

Параметр ParametersFile, имеющий aINI-тип «имя типизированного файла», применяется для определения списка дополнительных параметров, который также разрабатывается с использованием синтаксиса aINI. Шаблон представлен листингом 11, где [Additional parameters] – стандартная секция для размещения списка дополнительных шаблон-зависимых параметров (количество параметров произвольно); @paramname@ - имя параметра, @param\_value@ - значение параметра, @param\_description@ - описание параметра.

Листинг 11. Шаблон IMP файла дополнительных параметров в формате aINI

|  |
| --- |
| [Additional parameters]//Дополнительные шаблон-зависимые параметры  @paramname@=@param\_value@//@param\_description@  ... |

* + 1. **Репозиторий и база данных шаблонов**

Среди исходных данных задания на генерацию представлен параметр TemplateSID, имеющий aINI-тип «ссылка на запись в таблице базы данных» и определяющий используемый шаблон. Местоположение выбранного шаблона определяется на основе информации, размещенной в таблице базы данных gen.tmpls. Реляционная модель данных схематично представлена на рисунке 4б. Управление файловой библиотекой шаблонов было реализовано с использованием систем контроля версий (SVN, GIT).

На рисунке 4а представлена базовая структура каталогов, для хранения шаблонов. Регистрация шаблонов и их атрибутов, включая информацию о местоположении, а также хранение данных о преобразователях векторных и статических параметров, включая методы их замены, и прочая информация, хранится в таблицах схемы данных gen, входящей в состав базы данных РВС GCD подсистемы генерации кода (рисунок 4б).

|  |  |
| --- | --- |
|  |  |
| а) статическая структура каталогов  репозитория шаблонов с указанием  примерного местоположения шаблона GEN | б) таблицы базы данных схемы gen, обеспечивающие хранение атрибутов  шаблонов, включая их местоположение |

Рисунок 4 – Структура репозитория шаблонов и базы данных хранения их атрибутов и настроек генерации

Шаблоны сгруппированы по категориям, которые регистрируются в таблице tmcat (рисунок 4б). Регистрация отлаженных шаблонов осуществляется в таблице tmpls. Базовые статические параметры, которые могут быть использованы в любых шаблонах, регистрируются в таблице subst. Форматы выходных данных (листинг 1) при преобразовании векторных параметров регистрируются в таблице frmts. Каждому зарегистрированному формату соответствует конкретный преобразователь. Для определения преобразователей была использована технология XSLT. Применение конкретного XSL - преобразователя осуществляется автоматически при обработке каждого векторного параметра шаблона. Местоположение доступных XSL - преобразователей определяется на уровне ядра РВС GCD и выходит за рамки тематики настоящей статьи.

Параметр CodeObjectName используется для создания уникального имени формируемого объекта в совокупности с используемой структурой каталогов хранения и значениями параметров ComplexSID, SolutionSID, ProjectSID**.** Подробнее процесс регистрации объектов генерации в базе данных и алгоритм формирования их идентификаторов представлен далее.

* + 1. **Вложенные (зависимые) шаблоны**

На практике часто возникает необходимость повторного использования части кода шаблона в составе двух и более других шаблонов. В такой ситуации в рамках представляемого инструментария была реализована поддержка зависимых шаблонов согласно определению 4. На рисунке 5 представлены два шаблона: SFC и SFD, которые зависят от 6-ти других шаблонов: G7C, USP, DID, LCP, SA2, STT (рисунок 6). Реализация такой возможности позволила исключить дублирование кода при построении новых шаблонов. Количество уровней вложенности произвольно. Обработка исключительной ситуации потенциального формирования зацикливания при обработке шаблона не была реализована и является предметом будущих доработок созданного инструментария.

|  |  |
| --- | --- |
| 2018-08-03_19-03-57 | C:\Users\alsok_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\depended_template_06.png |
| а) SFC - Текст программы (ГОСТ 19.401-78): зависит от G7C, USP, DID, LCP, SA2, STT | |
| C:\Users\alsok_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\depended_template_02.png | C:\Users\alsok_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\depended_template_05.png |
| б) SFD - Описание программы (ГОСТ 19.402-78):зависит от G7C, USP, DID, LCP, SA2, STT | |

Рисунок 5 – Примеры двух шаблонов, определения которых зависят от других шаблонов

|  |  |
| --- | --- |
| C:\Users\alsok_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\depended_template_03.png | C:\Users\alsok_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\depended_template_04.png |
| а) G7C - Класс стиля ГОСТ 7.32-2001 | б) USP – Класс стиля «Единая система программной документации» (ЕСПД) |
| C:\Users\alsok_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\depended_template_09.pngC:\Users\alsok_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\depended_template_07.png | C:\Users\alsok_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\depended_template_10.pngC:\Users\alsok_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\depended_template_08.png |
| в) LCP, STT | г) DID, SA2 |

Рисунок 6 – Шаблоны, использующиеся в других шаблонах

* + 1. **Регистрация объектов, формируемых на основе шаблонов**

Создание объектов на основе шаблонов может приводить к формированию как корректных, так и некорректных результатов, что связано с существенным количеством входных параметров различных типов, значения которых определяются явно или неявно[[4]](#footnote-5).

Для отладки шаблонов и возможностей корректировки результатов генерации, помимо прочих, был предусмотрен параметр, определяющий путь к временному размещению результатов генерации (листинг 10, строка 9). После серии отладочных попыток генерации конечный удовлетворительный результат размещается в целевом каталоге хранения (листинг 10, строка 7). Следует заметить, что количество формируемых объектов, размещаемых в одних и тех же целевых каталогах, может быть достаточно велико. Это обстоятельство приводит к риску создания файла и/или каталога с именем, совпадающим с уже существующим объектом в целевом каталоге. Исключение такого риска может быть реализовано:

1. предварительным сканированием целевых каталогов на предмет существования файлов и/или каталогов с определёнными именами;
2. присвоением каждому генерируемому объекту на основе шаблона уникального имени (идентификатора) и размещением результата в целевом каталоге с присвоением имени, включающим этот идентификатор.

В рамках представленного подхода выбран второй вариант, это было связано со сложностью алгоритма генерации нового объекта и необходимостью размещения его в общем репозитории, независимо от количества формируемых объектов.

Для идентификации генерируемых объектов было предложено использовать строковые идентификаторы согласно синтаксису, представленному листингом 12.

Листинг 12. Синтаксис формата имени объекта, генерируемого на основе шаблона, в расширенной форме Бекуса-Наура с комментариями[[5]](#footnote-6)

|  |  |
| --- | --- |
| 1  2  3  4  5 | **FullCodeObjectName** = cpxSID slnSID "\_" tmlSID "\_" CodeObjectName.  **cpxSID** = ID3lc. (\**Строковый идентификатор комплекса (системы)\*)*  **slnSID** = ID3lc. (\**Строковый идентификатор решения (подсистемы)\*)*  **tmlSID** = ID3lc. (\**Строковый идентификатор шаблона\*)*  **CodeObjectName** = ID15CamelCase. (\**Наименование генерируемого объекта\*)* |

Такой синтаксис позволяет формировать идентификаторы, включающие семантику, что важно при обзоре сформированных материалов неподготовленным специалистом. Однако, иногда, для удобства поиска конкретного документа или объекта требуется информация о его коротком числовом идентификаторе. В рамках созданного программного инструментария такой идентификатор формируется только в случае сохранения результата генерации в едином репозитории. При этом сам идентификатор сохраняется вместе с соответствующим строковым идентификатором в специальную базу данных для удобства его последующего использования.

* 1. **Статическая и динамическая части структуры каталогов применяемых шаблонов**

Одним из важных приложений генератора кода является возможность заранее определить с помощью одного шаблона базовую структуру каталогов, а с помощью серии других шаблонов – структуры объектов, которые требуется размещать в этой заранее определённой базовой структуре. Отмеченные функциональные возможности реализуются путём разделения структур каталогов на статическую и динамическую как в шаблоне базовой структуры, так и в шаблонах структур размещаемых объектов.

На рисунке 7а представлена структура хранилища шаблонов: представлены четыре шаблона ART, PRS, PLG, USP, сгруппированные по трём категориям Research, Development и Styles. Шаблоны PRS и PLG зависят от USP. На рисунке 7б представлена структура каталогов, которая будет автоматически формироваться в результате генерации объектов на основе представленных шаблонов. Идентификаторы шаблонов выделены жирным шрифтом, названия категорий подчеркнуты.

|  |  |
| --- | --- |
| а) | б) |

Рисунок 7 – Пример структуры а) репозитория шаблонов и б) соответствующего ему автоматически формирующегося репозитория результатов генерации. Статические части шаблонов и соответствующая им статическая часть формирующегося репозитория подчеркнуты

Другими словами, статическая часть может быть сформирована заранее с помощью предварительно подготовленного шаблона базовой структуры. Динамическая часть может формироваться коллективом исследователей постепенно путём последовательной разработки отдельных документов и прочих материалов с использованием генератора кода.

1. **Программная реализация**

Программная реализация инструментария осуществлена на языке С++ и интегрирована в состав РВС GCD (рисунок 8).



Рисунок 8 – Принципиальная схема использования программных средств РВС GCD для интеграции созданного программного инструментария и его функционирования

Исходные данные для генерации (рисунок 8а) представлены в файле в формате aINI (листинг 10), что позволило автоматизировать создание графических меню пользователя для ввода исходных данных [25]. Для сохранения графовой модели (рисунок 8б) применялся язык описания графов aDOT [28]. Библиотеку функций генерации (рисунок 8в), исключая функции предикаты, определили функции, представленные в таблице 2. Программная реализация функций генерации кода осуществлена с использованием языка программирования С++. Числа, связанные с рёбрами схемы, показывают последовательность обработки данных.

Согласно представленной ранее архитектуре, программный инструментарий генерации кода может быть достаточно гибко дополнен новыми функциональными возможностями.

Применение aINI формата в качестве формата для файлов входных данных позволяет дополнять списки шаблон-зависимых параметров без изменения исходного кода генератора. Возможность хранения шаблон-независимых параметров в базе данных позволяет также дополнять их без внесения изменений в исходный код генератора.

1. **Примеры применения**

За время эксплуатации представляемого в настоящей статье программного инструментария были реализованы десятки шаблонов. В их числе: шаблоны файлов исходных данных для различных функций решения инженерных задач в рамках РВС GCD; шаблоны документов, форма представления которых должна соответствовать тем или иным нормативным документам или стандартам (в том числе государственным); шаблоны программных проектов и ряд других. Далее представлены несколько характерных примеров.

* 1. **Шаблоны файлов исходных данных**

Известным современным принципом разработки программного обеспечения является применение паттерна проектирования «модель-представление-контроллер» (англ. «model-view-controller» или чаще просто MVC). Данный принцип используется в рамках применения объектно-ориентированного подхода и предполагает жесткую типизацию классов, создаваемых при проектировании и разработке ПО. Для обеспечения независимости классов «представлений» от классов «моделей» может применяться файл в едином формате, который будет использоваться одновременно и теми, и другими, однако по-разному. «Представления» должны интерпретировать файлы в этом формате и визуализировать данные, тогда как «модели» должны использовать эти файлы для загрузки данных и их обработки.

Предметом разработки в инженерном ПО, как правило, являются функции, необходимые для решения задач, реализующих те или иные вычислительные операции. Каждая из таких функций требует определения своего множества входных параметров (исходных данных), которые, в свою очередь, должны быть формализованы. Формализация исходных данных может быть обеспечена, например, применением шаблонизируемых текстовых форматов данных (рисунок 9).



Рисунок 9 – Применение шаблонов файлов исходных данных

В РВС GCD в качестве такого формата данных используется aINI [25]. Среди известных и широко используемых форматов для решения аналогичных задач можно выделить: JSON, XML.

Примеры шаблонов, подготовленных в формате aINI, представлены листингами 13, 14, 15. Формат, представленный листингом 13, использовался для определения исходных данных в задаче построения геометрической модели представительного элемента объёма (ПЭО) монодисперсного композиционного материала (КМ) на базе сферических включений в рамках численного анализа эффективных упругих характеристик дисперсно-армированных КМ [29]. Форматы, представленные листингами 14, 15 использовались в рамках решения задачи оптимизации топливных затрат ТЭЦ [30].

Листинг 13. Шаблон файла *@DocSID@.fbr* исходных данных в формате aINI для генерации ПЭО дисперсно-армированных КМ

|  |
| --- |
| **[Input data]** // Исходные данные  \***OutputGeoName**=@OutputGeoName@ // Наименование формируемой геометрической модели  -**OutputGEOFile**=@GeneratedDate@\_@OutputGeoName@.geo // Имя файла выходной геометрии  \***GenerateMesh**=[1]{0|1} // Генерировать расчетную сетку  **MeshGranularity**=[@MeshGranularity@]{verycoarse|veryfine|moderatde|fine|moderate|coarse} //Качество расчетной сетки  -**OutputAneuFile**=@GeneratedDate@\_@OutputGeoName@\_@MeshAttrs@.aneu //Имя файла расчетной сетки  // Абсолютные размеры ячейки периодичности  -**WidthX**=@WidthX@ [[mm]] // Ширина  -**HeightY**=@HeightY@ [[mm]] // Высота  -**LengthZ**=@LengthZ@ [[mm]] // Длина  \***Concentration**=@Concentration@ // Концентрация наполнителя  \***Radius**=[@CurrentR@;@MinR@:@MaxR@;@Step@] // Внешний радиус включения |

Листинг 14. Шаблон файла *@DocSID@.int* исходных данных для программного модуля аппроксимации расходной характеристики одного энергоблока ТЭЦ

|  |
| --- |
| [**Input data**]# Исходные данные для построения интерполирующей зависимости  **InputFile**=[@InputFile@.csv] // Файл с опорными точками (исходные данные)  **ExpIntData**=[@ExpIntData@] // файл для записи коэффициентов интерполяции  **smoothing**=[1]{0|1} // параметр сглаживания |

Листинг 15. Шаблон файла *@DocSID@.gvl* файла исходных данных для программного модуля вычисления значения расхода топлива при произвольных электрической нагрузке и отпуску тепла на основе ранее построенной расходной характеристики для одного энергоблока ТЭЦ

|  |
| --- |
| [**Input data**]// Вычисление значения суммарного расхода топлива на энергоблоке  **ExpIntData**=[@ExpIntData@]//Интерполированная расходная характеристика блока  **InputFile**=[@InputFile@.csv]// Файл с опорными точками (исходные данные)  **Ne**=@Ne@ [[MWt]]//Электрическая нагрузка блока  **Qt**=@Qt@ [[GkalH]]//Отпуск теплоты блоком посредством ПСГ, Гкал/ч |

Представленные шаблоны использовались как основа для формирования графического пользовательского интерфейса [25], далее для формирования файла исходных данных, сохранения введённых значений параметров и их дальнейшей загрузки функциями обработки данных.

* 1. **Шаблон документации на программную реализацию вычислительного метода**

Одной из прикладных программных разработок, формализующих процесс описания алгоритма вычислительного метода, является графоориентированный подход [27]. Программная реализация вычислительного метода (создание «решателя» задачи) с его помощью сводится к построению графовой модели метода, определению файла исходных данных и реализации множества функций перехода [28].

Применение формализованной процедуры разработки позволило создать шаблон SLR документации на произвольный реализуемый таким образом метод. На рисунке 10 представлен пример использования шаблона SLR (листинг 16) генератором кода, при подготовке документации по одному из разрабатываемых решателей. На рисунке выделены некоторые подставленные значения: красным – скалярных параметров, синим – значения векторных параметров, зелёным – значения скалярных функциональных параметров.

|  |  |
| --- | --- |
| C:\Users\alsok_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2018-08-14_23-16-15.png | C:\Users\alsok_000\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2018-08-14_23-19-48.png |
| а) первая страница | б) последняя страница |

Рисунок 10 – Фрагменты автоматически сгенерированной документации в формате PDF по графоориентированной программной реализации метода решения задачи обратной гомогенизации упругих характеристик композиционных материалов на основе шаблона SLR, подготовленного на языке вёрстки LaTeX

Листинг 16 Фрагмент шаблона SLR

|  |
| --- |
| Описание инструмента решения задачи (решателя) \textbf{**@Solver.slver@**}.  \begin{tabular}{|>{\raggedright}p{0.4\linewidth}|p{0.6\linewidth}|} \hline  Идентификатор инструмента решения задачи (решателя) & \textbf{**@Solver.slver@**} \\ \hline  Описание решателя & **@Solver.dscra@** \\ \hline  Идентификатор связанной сетевой модели (графа) & \textbf{**@Solver.ntmdl@**}\\  \hline  \end{tabular}  \subsection{Зависимые решатели от **@Solver.slver@**}  {\small **%#com.sub\_slvrs\_v2[par\_slver='@Solver.slver@']:TEX#%**}  \begin{landscape}  \subsection{Описания функций решателя @Solver.slver@}  На рисунках \ref{fig:**@Solver.slver@**}, \ref{fig:**@Solver.slver@**\_full} представлены соответственно краткая и полная сетевые модели решателя \textbf{**@Solver.slver@**}. Описания каждой функции обработки данных и функций-предикатов представлены в таблице ниже. |

* 1. **Шаблон отчета о проведённых патентных исследованиях согласно ГОСТ Р 15.011-96**

В рамках большинства научно-исследовательских работ, проводимых в России в рамках государственных заказов, на отдельных этапах их реализации возникает необходимость проведения патентных исследований в соответствии с требованиями ГОСТ Р 15.011-96. Еще одним примером прикладного использования разработанного программного инструментария стала задача автоматизации построения отчета о проведённых патентных исследованиях.

В зависимости от задачи, патентные исследования подразделяют на исследования: тенденций и уровня развития объекта хозяйственной деятельности; патентоспособности; патентной чистоты; с представлением карты патентного ландшафта.

Построение отчета о проведенном патентном исследовании первого вида (анализ тенденций и уровня развития объекта хозяйственной деятельности) может быть автоматизировано в связи с незначительным объёмом неформализуемого материала, включенного лишь в раздел, связанный с анализом найденных патентных документов, и в заключение. Таким образом был разработан шаблон, позволяющий генерировать отчет о проведенном патентном исследовании первого из отмеченных выше видов в соответствии с требованиями ГОСТ Р 15.011-96 с использованием языка вёрстки LaTeX. Источником данных стала разработанная в рамках РВС GCD подсистема проведения патентных исследований.

С использованием разработанного шаблона были подготовлены отчеты о патентных исследованиях в рамках серии государственных и хоздоговорных НИР и ОКР, выполненных в МГТУ им. Н.Э. Баумана, участниками которых были авторы. В результате применения генератора кода и созданного шаблона удалось обеспечить автоматическое построение каркаса отчета и наполнить его данными. При этом были автоматизированы процессы, позволяющие в среднем сформировать более 90% всего содержания отчетов. Отчеты прошли успешную проверку в организациях-мониторах.

**Заключение**

Созданный программный инструментарий лёг в основу автоматизированной методологии разработки программного обеспечения инженерного анализа, разрабатываемого авторами. Анализ результатов проведенных ими поисковых исследований и выполнения прикладных работ позволяет сделать следующие выводы.

* Представляется целесообразным при разработке программного обеспечения инженерного анализа применение «гибридных» подходов, а именно: для разработки общесистемных стандартных функциональных возможностей следует использовать генераторы кода на основе шаблонов, тогда как при разработке программных реализаций алгоритмически сложных вычислительных процедур следует использовать генерацию кода на основе графовых представлений алгоритмов [28], что является предметом текущих разработок.
* Применение средств поддержки процессов разработки и средств генерации кода позволяет систематизировать процесс разработки многофункциональных программных комплексов. Представляется обоснованным применение таких средств при создании прикладного наукоёмкого программного обеспечения.
* Актуальными и востребованными на практике являются программные механизмы предоставления удалённого доступа к разработанному программному инструментарию и библиотеке шаблонов посредством web-приложения. Решение такой задачи позволит сформировать основу средств автоматизации процессов разработки программного обеспечения, документирования, безбумажного документооборота, а также позволит предоставить доступ к созданному инструментарию широкому кругу пользователей. В работах по созданию web-приложения участвует студент Лаишевский И.А., который подключился к развитию программного инструментария с 2018 года.

Разработка представленного в настоящей статье программного инструментария выполнена авторами в инициативном порядке в МГТУ им. Н.Э. Баумана при разработке ядра «Распределенной вычислительной системы GCD».

**Список литературы**

1. Орлов C. Технологии разработки программного обеспечения: Учебник – СПб.: Питер, 2002. – 464 с.: ил. ISBN 5-94723-145-Х.
2. Лаврищева Е.М. Software Engineering компьютерных систем. Парадигмы, технологии и CASE-средства программирования. - К.: Наукова думка, 2013. - 283 с. - ISBN 978-966-00-1416-9.
3. Jörges S. Construction and evolution of code generators 7747. Ch 2 The state of the art in code generation. Berlin Heidelberg: Springer; 2013. pp. 11–38.
4. Syriani, E., Luhunu, L., Sahraoui, H. Systematic mapping study of template-based code generation. Computer Languages, Systems and Structures. Volume 52, June 2018, Pages 43-62.
5. Rosales-Morales, V.Y., Alor-Hernández, G., García-Alcaráz, J.L., Zatarain-Cabada, R., Barrón-Estrada, M.L. An analysis of tools for automatic software development and automatic code generation. Revista Facultad de Ingenieria. Issue 77, 2015, Pages 75-87.
6. Федотова Д.Э., Семенов Ю.Д., Чижик К.Н. CASE-технологии. – М.: Горячая линия – Телеком, 2005. – 160 стр. – ISBN 5-93517-121-х.
7. Lúcio L., Amrani M., Dingel J., Lambers L., Salay R., Selim G.M., et al. Model transformation intents and their properties. Softw. Syst. Model 2014; Vol. 15, No. 3, pp.685 – 705.
8. Fleischer, D., Beine, M., Eisemann, U. Applying model-based design and automatic production code generation to safety-critical system development. SAE International Journal of Passenger Cars - Electronic and Electrical Systems. Volume 2, Issue 1, 2009, Pages 240-248.
9. Wikipedia. The Free Encyclopedia. Comparison of code generation tools [Электронный ресурс].

URL: <https://en.wikipedia.org/wiki/Comparison_of_code_generation_tools> (дата обращения: 13.07.2018).

1. Lechanceux Luhunu and Eugene Syriani. Comparison of the Expressiveness and Performance of Template-Based Code Generation Tools. SLE’17, October 23–24, 2017, Vancouver, Canada, pp. 206-216, DOI: 10.1145/3136014.3136021
2. Kshitija Shinde, Yu Sun. Template-Based Code Generation Framework for Data-Driven Software Development. Proceedings of 2016 4-th International Conference on Applied Computing and Information Technology (ACIT 2016), December 12-14, 2016, University of Nevada, Las Vegas, USA, pp. 55-60, DOI: 10.1109/ACIT-CSII-BCD.2016.023.
3. Andersson, P. and Höst, M. UML and SystemC - A Comparison and Mapping Rules for Automatic Code Generation, in Embedded Systems Specification and Design Languages, E. Villar (ed.). Amsterdam, Netherlands: Springer, 2008, pp. 199-209.
4. Wehrmeister, M.A., De Freitas, E.P., Binotto, A.P.D., Pereira, C.E. Combining aspects and object-orientation in model-driven engineering for distributed industrial mechatronics systems. Mechatronics. Volume 24, Issue 7, 1 October 2014, pp. 844-865.
5. Sasi Bhanu, S.J., Vinaya Babu, A., Trimurthy, P. Code generation for semantic evolution of embedded systems. ARPN Journal of Engineering and Applied Sciences. Volume 10, Issue 20, 2015, pp. 9382-9394.
6. Деев Д.В., Окуловский Ю.С Система порождения документов в форматах HTML, MHT и WordProcessingML. Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2010. № 6, Т.70, С. 77-81.
7. Krzysztof Kołek and Krzysztof Piątek. Rapid algorithm prototyping and implementation for power quality measurement. Journal on Advances in Signal Processing. 2015. Vol.19, pp. 1-12. DOI 10.1186/s13634-015-0192-3
8. Kai Hu, Zhangbo Duan, Jiye Wang, Lingchao Gao, Lihong Shang. Template-based AADL automatic code generation. Frontiers of Computer Science. Volume 13, Issue 4, 2019, pp. 698-714. DOI: 10.1007/s11704-017-6477-y
9. Александров А.Е., Шильманов В.П. Инструментальные средства разработки и сопровождения программного обеспечения на основе генерации кода. Бизнес-информатика. 2012. № 4, Т.22, С. 10-17.
10. Самохвалов Э.Н., Ревунков Г.И., Гапанюк Ю.Е. Генерация исходного кода программного обеспечения на основе многоуровневого набора правил. Вестник Московского государственного технического университета им. Н.Э. Баумана. Серия: Приборостроение. 2014. № 5, Т.98, С. 77-87.
11. Софин Н.A., Кавалеров М.В. Среда разработки для автоматизированной генерации исходного кода и документации. Master’s Journal. 2016. № 2. С. 275-262.
12. Соколов А.П., Шпакова Ю.В., Першин А.Ю. Проектирование распределенной программной системы GCD численного моделирования композитов. // Математические методы в технике и технологиях - ММТТ-25: сб. трудов XXV Международной научной конференции: в 10 т. T. 5. Секция 8,9 / под. общ. ред. А.А. Большакова. – Волгоград: ВолГУ, 2012, с.79-80.
13. Соколов А.П., Макаренков В.М., Шевцов А.С. [Разработка программы gcdcli\_plg\_CodeGenerator автоматизации процессов формирования электронных документов](http://conf.nsc.ru/ym2013/ru/reportview/175636) // Материалы XIV Всероссийской конференции молодых ученых по математическому моделированию и информационным технологиям, - г. Томск, Россия, Томский филиал Института вычислительных технологий СО РАН, 14 - 18 октября 2013, с. 46-47.
14. Свид. 2014612782 Российская Федерация. Свидетельство об официальной регистрации программы для ЭВМ. Программа gcdcli\_plg\_CodeGenerator для автоматического формирования документов и программных объектов на основе шаблонов / Соколов А.П., Макаренков В.М.; заявитель и правообладатель: Соколов А.П., Макаренков В.М. (RU). – № 2013617477, заявл. 07.08.2013, опубл. 06.03.2014, Реестр программ для ЭВМ. - 1 с.
15. Соколов А.П., Першин А.Ю. Программный инструментарий для создания подсистем ввода данных при разработке систем инженерного анализа – Программная инженерия. 2017. Т. 8, № 12, с. 543-555.
16. Димитриенко Ю.И., Соколов А.П. Система автоматизированного прогнозирования свойств композиционных материалов // Информационные технологии. 2008. № 8. С. 31–38.
17. Патент на изобретение RU 2681408. Способ и система графо-ориентированного создания масштабируемых и сопровождаемых программных реализаций сложных вычислительных методов / Соколов А.П., Першин А.Ю.; заявитель и правообладатель: Соколов А.П., Першин А.Ю. (RU). – заявка № RU 2017 122 058 A, приоритет 22.07.2017, опубликовано 22.02.2019, Реестр изобретений Роспатента. – 1 стр.
18. Соколов А.П., Першин А.Ю. Графоориентированный программный каркас для реализации сложных вычислительных методов. Программирование. 2019, T.47, №5, c. 43-55.
19. Соколов А.П., Щетинин В.Н. Идентификация упругих свойств адгезионного слоя дисперсно-армированных композитных материалов на основе экспериментальных данных - Механика композиционных материалов и конструкций – T.24, №4 – 2018, c. 555-581.
20. Соколов А.П., Щетинин В.Н., Сапелкин А.С. Применение технологии GBSE для разработки программного обеспечения автоматизации процесса распределения электрических и тепловых нагрузок на ТЭЦ \\ Материалы XX Юбилейной международной конференции по вычислительной механике и современным прикладным программным системам (ВМСППС’2017), 24 – 31 мая 2017 г., Алушта. – М: Изд-во МАИ, 2017 – С. 108-110.

Соколов А.П., Макаренков В.М., Першин А.Ю., Лаишевский И.С. Разработка программного обеспечения генерации кода на основе шаблонов при создании систем инженерного анализа

**Содержание**

**ВВЕДЕНИЕ**

**1. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ИНСТРУМЕНТАРИЯ**

1.1. Базовые понятия

1.2 Формат файлов исходных данных aINI

1.2 Использования параметров шаблонов векторного типа

1.3 Использование входных параметров, имеющих aINI-тип ссылка на запись в таблице удаленной базы данных

1.4 Архитектура и состав программного инструментария

1.4.1 Особенности программной реализации алгоритма генерации

1.4.2 Формат файла исходных данных задания на генерацию

1.4.3 Репозиторий и база данных шаблонов

1.4.4 Вложенные (зависимые) шаблоны

1.4.5 Регистрация объектов, формируемых на основе шаблонов

1.5. Статическая и динамическая части структуры каталогов шаблонов

**2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

**3. ТЕСТИРОВАНИЕ**

3.1 Шаблон документации на графоориентированный решатель задач.

3.2 Шаблон отчета о проведённых патентных исследованиях согласно ГОСТ Р 15.011-96.

**ЗАКЛЮЧЕНИЕ**

**А.П. Соколов**1,**\***, канд. физ.-мат. нацк, доц., E-mail: [alsokolo@bmstu.ru](mailto:alsokolo@bmstu.ru)

**В.М. Макаренков**1, студент

**А.Ю. Першин**1, ассистент, E-mail: [tony.pershin@gmail.com](mailto:tony.pershin@gmail.com)

**И.С. Лаишевский**1, бакалавр, E-mail: [ivanlaish@yandex.ru](mailto:ivanlaish@yandex.ru)

1Московский Государственный Технический Университет имени Н.Э. Баумана, Москва, Россия

**Разработка программного обеспечения генерации кода на основе шаблонов при создании систем инженерного анализа**

*Генерация кода на основе шаблонов предполагает автоматизированное формирование исходного кода программ или некоторого текстового результата, соответствующего заранее определённому формату, называемому шаблоном. Представлен программный инструментарий автоматизированного прототипирования программных модулей и документов различного типа и назначения. В основе созданных технических решений лежит применение разработанных авторами: специализированного формата представления шаблонов, формата aINI файлов исходных данных и библиотеки функций, позволяющей интерпретировать шаблоны и создавать на их основе объекты различных типов. Представлены алгоритм генерации и метод автоматической интеграции созданного программного инструментария в рамки программного комплекса «Распределённая вычислительная система GCD». Приведены примеры практически значимых шаблонов с описанием их применения. В их числе: шаблоны файлов исходных данных; шаблон документации на программную реализацию сложного вычислительного метода; шаблон отчета о проведённых патентных исследованиях согласно ГОСТ Р 15.011-96.*

***Ключевые слова:*** автоматизация процессов разработки; быстрое прототипирование программного обеспечения и документации; template-based code generation; model-driven engineering; model-to-text transformation; aINI; генерация кода; автоматизация программирования; интерпретация текстовых данных; программные процессоры; технологии разработки систем инженерного анализа.

1. Известные CASE инструментарии: системы управления конфигурациями: контроля версий (SVN, GIT, Mercurial и пр.), построения инсталляторов, непрерывной интеграции (CI); средства управления требованиями; средства планирования (например, Redmine, Jira, MS SharePoint и др.); средства анализа программ (анализаторы программ); средства тестирования, отладки, документирования; системы автоматизации построения графических пользовательских интерфейсов; системы автоматизации генерации кода; и др. [↑](#footnote-ref-2)
2. Например, если шаблоном определён выходной язык, то возможно получение ошибочно формируемого кода при синтаксически несоответствующих подстановках вычисленных значений параметров, используемых в шаблоне. Такое поведение возможно в случае модификации функции вычисления значения параметра либо при неверном изменении значения параметра. [↑](#footnote-ref-3)
3. Разработка наукоёмкого ПО представляет самостоятельную сложность, а с усложнением применяемых математических моделей, вычислительных методов и развитием вычислительных систем, актуальность применения новых прогрессивных методов создания ПО будет лишь расти. [↑](#footnote-ref-4)
4. Например, шаблон PAT, разработанный с использованием созданного программного инструментария, предназначенный для формирования отчета о патентных исследованиях согласно ГОСТ Р 15.011-96, требует определения 37 скалярных дополнительных параметров, среди которых много тех, чьи значения определяются во время выполнения. [↑](#footnote-ref-5)
5. Для краткости опущены определения нетерминальных символов ID3lc (строковый идентификатор длины 3, определённый в нотации lc), ID15CamelCase (строковый идентификатор длины 15), определённый в нотации CamelCase (см. обозначения в таблице 1). [↑](#footnote-ref-6)