Разработка программы ведения полевой геологической документации

С. А. Беляев

Санкт-Петербургский государственный электротехнический университет (ЛЭТИ)
Санкт-Петербург, ул. Профессора Попова, 5

e-mail: bserge@bk.ru

Аннотация. Полевая документация в большинстве случаев ведется в бумажном виде с использованием карандаша, в редких случаях используется лицензионное программное обеспечение. Это связано с высокой стоимостью оборудования и основными финансовыми вложениями именно в оборудование и технологии бурения, а не средства документирования. В работе предлагается описание построения универсальной программы ведения полевой геологической документации, которая решает существенную часть задач своих конкурентов в части ведения полевого журнала. В работе описаны основные требования к программе ведения геологической документации, сформулированы ее ограничения, предложены решения по каждой из ключевых задач разработки, приведены принципиальные фрагменты программного кода на Java. Описаны подходы к формированию текстового описания для каждого интервала с использованием скриптового языка JavaScript, в том числе с использованием данных по подынтервалам.
Ключевые слова: полевая геологическая документация, разработка программного обеспечения, Java, JavaScript.

1. Введение

Геологическая документация включает длинный список документов, таких как полевые книжки, полевые журналы документации горных выработок, каталоги горных выработок и буровых скважин, различные акты и этикетки и многое другое [1]. В соответствии с учебным пособием [1] и [2] заполнять полевую документацию необходимо исключительно простым карандашом, ошибки исправляются зачеркиванием и правильным написанием, помарки и исправления «цифра по цифре» не допускаются [3]. Особое внимание уделяется зарисовкам естественных и искусственных обнажений, отбору и документации образцов [4]. В соответствии с [5] при зарисовках в документации следует использовать единые условные обозначения (ГОСТ 2.857-75), и заполнять документацию следует в строго определенном порядке [3] в соответствии с рекомендованными формами документов с ведением и сохранением геологических материалов.

За весь комплект документации по месторождению отвечает геолог или старший геолог, а оформляют документацию техники-геологи (гидрогеологи) [5, 6].

При большом объеме проходимых выработок (канав, шурфов, скважин) у старшего геолога не всегда есть возможность проверить качество заполнения всей документации [5].

При документировании скважины выполняются [6]:

1) описание горных пород каждого слоя или рейса;
2) выделение и детальное описание интервалов распространения полезных ископаемых, их прямых и косвенных признаков, выделение и описание интервалов распространения потенциально продуктивных пород (далее — интервалов);
3) описание характера границ слоя с выше- и нижележащими образованиями;
4) измерение угла наклона каждого слоя к оси керна;
5) измерение мощности каждого слоя вдоль оси керна;
6) описание трещиноватости керна;
7) фиксация плоскостей притирания;
8) сбор ископаемых органических остатков, описание их расположения по отношению к слоистости или оси керна;
9) отбор образцов и проб.

В [6] отмечается, что основной формой документации скважин является полевой журнал геологической документации скважин, который может изменяться в зависимости от специфики изучаемого района.

Не сложно представить себе заполнение геологических документов простым карандашом даже в случае требований по двойному документированию каждого керна буровой скважины [6], но с учетом развития современных технологий можно предположить наличие соответствующих программных продуктов, упрощающих сложный процесс документирования. При этом можно было бы рассмотреть использование электронных таблиц, таких как Microsoft Excel, которые являются универсальным и очень гибким инструментом. Однако для создания шаблонов с учетом особенностей месторождений, последующего контроля ошибок при их заполнении и автоматизированного формирования необходимых текстовых описаний для каждой строчки документации необходимо написание соответствующих программ (макросов). Следует отметить, что геологическая практика требует качественных текстовых описаний. Данная задача может быть решена при использовании VBA для Microsoft Excel и сохранения необходимых данных в Microsoft Access. Но такой подход, с одной стороны, ничем не отличается от использования любого другого языка программирования (отобразить таблицу можно не только в Microsoft Excel), при этом существенно ограничит в возможностях использования

операционных систем и требует наличия на каждом компьютере соответствующих лицензий Microsoft.

При поиске программ ведения геологической документации в сети Интернет на русском языке с легкостью находятся шаблоны и формы для текстовых редакторов, но не так просто найти подходящее программное обеспечение. При поиске на английском языке (geologic logging) можно найти программные продукты самого различного качества: от бесплатно распространяемых, таких как SGS-Geobase (www.geostat.com), Geotriple for geospatial imagery (www.geoforge.org), до серьезных коммерческих продуктов, таких как Sedlog (www.sedlog.com), Georeka (www.georeka.com), Geology RockWare (www.rockware.com). Можно найти и российские решения, такие как «Информационная система АГР» [7].

В [2] сформулированы причины медленного развития программ документирования, приведена критика существующих программ ведения геологической документации, сформулированы основные требования к программному обеспечению, необходимому специалистам-геологам. Следует отметить, что до настоящего времени не все из заявленных в [2] требований нашли свое отражение в реальных программных продуктах.

2. Постановка задачи

Из практики ведения полевой геологической документации старший геолог при выборе инструмента работы помимо основных требований обращает внимание на следующие возможности программного продукта:

1) его стоимость — основной бюджет вкладывается в оборудование, обычно не очень много средств выделяется на другие нужды;
2) функциональные возможности должны соответствовать возлагаемым на продукт задачам — например, техникам-геологам не нужен функционал по 3D-представлению месторождения, в том числе из-за вопросов коммерческой тайны;
3) поддержка нескольких языков — это связано с интернациональностью команд, проводящими изыскания;
4) возможность простой, но гибкой настройки — добавление, удаление и изменение элементов документации на уровне шаблона, создаваемому старшим геологом, в том числе по составу справочников, чтобы техник-геолог даже не видел элементов документации, не соответствующих месторождению;
5) максимальная защита от ошибок ввода — все данные справочников должны выбираться из справочников, автоматически заполняемые поля

должны заполняться в соответствии с предопределенными формулами, в том числе нужны подсказки при потенциальных ошибках заполнения;

6) максимально простой и распространенный язык преобразования табличных данных в текстовое описание — изменения в правила преобразования вносятся не часто, поэтому не удобно, когда языком преобразования в полной мере владеет только разработчик продукта;
7) возможность работы под управлением различных операционных систем.

Критерию поддержки в том числе русского языка не удовлетворяет большинство зарубежных и тем более бесплатных продуктов. У бесплатных продуктов к тому же наблюдается проблема с гибкостью настройки шаблонов документации и с формированием текстовых описаний как таковых. Серьезные коммерческие решения, такие как «Информационная система АГР» [7], поддерживают несколько языков, обеспечивают гибкую настройку шаблонов, обеспечивают защиту от ввода некорректных данных, но описания формируют с использованием скриптового языка собственной разработки и не позволяют в полной мере обеспечить «защиту от дурака» для отдельных, наиболее сложных по заполнению листов геологической документации (таких как «Керновое опробование»). Разработчик «Информационная система АГР» выбрал путь пересчета перечня контрольных проб по команде, а не в автоматическом режиме в процессе заполнения, что приводит к влиянию человеческого фактора на качество создаваемой документации. Таким образом, наблюдается недостаток в простом, но достаточно гибком решении для полевой работы.

В большинстве современных программ имеется логичное разделение функционала ведения полевой документации и создания шаблона данной документации. Следует отметить, что шаблоны в целом похожи, но в деталях отличаются от месторождения к месторождению. На старшего геолога, отвечающего за исследование месторождения, возлагается задача разработки шаблона, которым впоследствии будут пользоваться техники-геологи.

С точки зрения функциональных требований редактор шаблонов программы ведения полевой геологической документации должен обеспечивать:

- описание перечня листов геологической документации;
- ведение перечня и значений справочников (с учетом языков);
- описание правил формирования текстовых описаний в геологической документации (с учетом языков);
- редактор параметров шаблона.

Функциональные требования к редактору полевой геологической документации:

- ввод данных в соответствии с шаблоном;

– обновление шаблона (данная потребность может возникнуть, если в шаблоне изначально не предусмотрели все особенности месторождения);
– экспорт данных в стандартных форматах.

3. Подход к решению

3.1. Описание в виде «дерева»

Возможны два подхода к созданию: специализированное или универсальное решение. Специализированный подход учитывет все особенности формирования документации, но не позволит гибко вносить изменения без привлечения программистов. Соответственно, рассмотрим универсальный вариант. При этом вся документация разделяется на листы, в каждом листе — таблица, а в рамках шаблона необходимо описать таблицу и правила ее заполнения. Учитывая большой объем информации и необходимость ее детальной группировки, предлагается предусмотреть иерархическую структуру столбцов, когда столбцы могут группироваться. Следует отметить, что группировки предусмотрены и в коммерческих решениях (https://agr4.ru/).

Учитывая иерархичную структуру листов и входящих в них полей, целесообразно описание состава геологической документации выполнять с использованием «деревьев». В качестве листьев будут выступать столбцы, в которые оператор будет вводить данные, а первыми узлами от корня будут таблицы.

При редактировании «дерева» должны быть предусмотрены традиционные для этого функции: добавление, редактирование, удаление и перемещение узла дерева. У каждой таблицы и каждого узла есть имя на тех языках, которые предусмотрены решением. Листья «дерева» представляют собой столбцы таблицы, которые могут быть различных типов. Практика показывает, что при формировании документации используются следующие типы данных: символьные строки (текст), целые и вещественные числа, значения из справочников. При описании шаблона достаточно указать, из какого справочника будет заполняться значение данного столбца, а уже в редакторе будет выбираться конкретное значение.

3.2. Справочники

Сами справочники в большинстве случаев представляют собой список из наименований на нескольких языках. В итоге при отображении в редакторе пользователю может отображаться «выпадающий список», из которого он выберет интересующее его значение.

3.3. Именование

Следует отметить, что листы и столбцы в геологической документации имеют различное значение, при этом выделяют такие листы, как, например, «Вмещающие породы», которые формируют общее описание для заданных интервалов глубин, «Подчиненные породы», «Минерализация», которые содержат подынтервалы в рамках общих интервалов, «Керновое опробование», которое содержит специфическую информацию о сделанных пробах. При этом связь между таблицами для определения соответствия строчек осуществляется по интервалам, соответственно, для автоматического анализа в рамках программы они должны называться одинаково, например «FROM» и «TO». Учитывая универсальность описываемого решения имена «от» и «до» для интервала, имена таблиц со специализированными правилами обработки должны задаваться в редакторе параметров шаблона. Следует отметить, что имена столбцов «от» и «до» для интервала при необходимости сопоставления между листами должны быть одинаковыми.

При этом все программисты знают, что оперировать именами, да еще и с учетом различных языков — «плохой» стиль программирования. Это то же самое, что в базе данных в качестве первичного ключа использовать ФИО, а не отдельно введенный идентификатор. Но в данном случае использование числовых полей не удобно, поэтому целесообразно для каждой таблицы и каждого поля ввести символическое имя, чтобы к нему можно было обращаться в том числе при выполнении правил заполнения полей и формировании описаний. Предлагается при этом использовать ограничение на символические имена, например только латинскими буквами в верхнем регистре и цифрами (данное ограничение будет обосновано дальнейшим изложением). При этом используемые символьные имена должны быть уникальны в рамках одного листа, чтобы пользователю не нужно было запоминать десятки имен, которые он использовал в других листах.

3.4. Формулы

Универсальность накладывает еще одно ограничение — необходимость описания правил заполнения полей. Простейшие примеры правил.

1. Увеличить на единицу по сравнению со значением в предыдущей строке.
2. В поле «от» текущей строки вписать значение поля «до» предыдущей строки.
3. Вычислить «до» на основании «от» и заданной «длины».

Использование правил позволит избежать ошибок заполнения документации, но на этапе формирования шаблона требует от оператора знания некого языка, ко-

торый позволит описывать эти правила. Традиционно есть два подхода: изобретение нового языка либо использование существующего скриптового языка. Целесообразно рассмотреть возможность использования готового скриптового языка. С учетом нефункциональных требований в качестве основного языка разработки может быть выбран язык Java, при этом разработка интерфейса может осуществляться на современном JavaFX, а в качестве скриптового — целесообразно использовать JavaScript, который имеет низкий порог вхождения для новичков. Использование JavaScript обусловлено движком V8 Nashron, реализованном в Java. В JDK 8 он поддерживает стандарт ECMAScript 5, но для описания формул и правил в рамках геологической документации это не является ограничением, мало того, это является преимуществом по сравнению с ограниченными «самодельными» скриптовыми языками.

При этом при описании правил предлагается использовать следующее именование полей: символическое имя листа, за которым добавить знак подчеркивания и затем — символическое имя поля.

В примере

```
HR_TO = HR_FROM + HR_LENGTH;
```

лист называется «HR», поле «до» называется «TO», поле «от» называется «FROM», поле «длина» называется «LENGTH». При этом целесообразно привязать данную формулу к соответствующему полю, чтобы в том числе была возможность автоматически обнаружить, что введение текста в это поле невозможно (рис. 1).

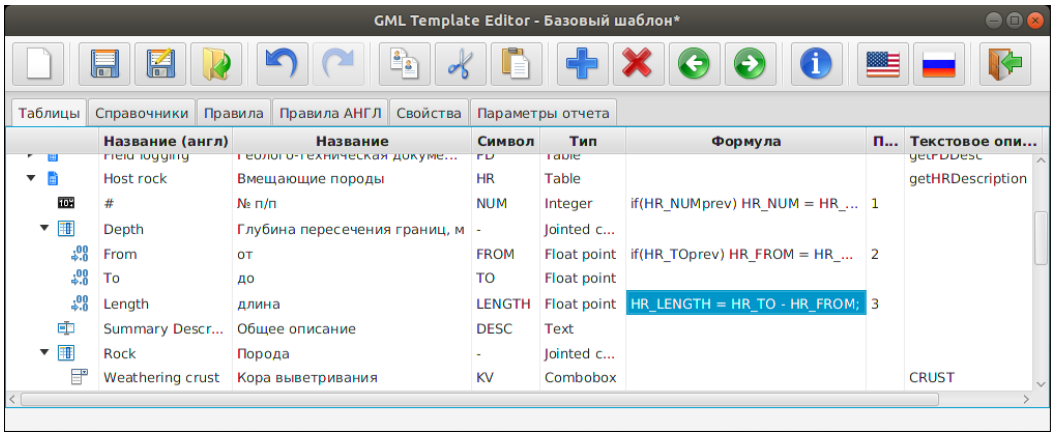

Рисунок 1. Редактирование формул

Изучая приведенные примеры правил, видно, что необходимо описать способ обращения к предыдущей строке. Это предлагается делать, добавляя постфикс «prev» к переменным. Здесь важно, что постфикс пишется маленькими буквами,

что позволяет отличать его от символьных имен полей и выполнять обработку в автоматическом режиме.

В строке

```
if(HR_TOprev) HR_FROM = HR_TOprev; else HR_FROM = 0;
```

проверяется значение поля «TO» в предыдущей строке (prev), если (if) оно задано, то это значение записывается в поле «FROM», иначе (else) в него записывается ноль.

Строка

```
if(HR_NUMprev) HR_NUM = HR_NUMprev + 1; else HR_NUM = 1;
```

означает, что если задано предыдущее значение поля «NUM», то в текущей строке оно будет увеличено на единицу, иначе будет задано начальное значение «1».

При этом важно, что в ситуации, когда в листе будет множество формул (минимум две), которые зависят друг от друга, то может возникнуть ситуация, в которой важна последовательность вычисления формул. Соответственно, программа должна предусматривать задание последовательности вычисления.

3.5. Правила создания описания

Правила полевого описания являются вполне сформировавшимися и существуют общепринятые подходы для различных языков. Нарушение последовательности в тексте затрудняет восприятие и может приводить к ошибкам. Конечно, могут использоваться подходы для автоматического анализа текста на естественном языке с учетом элементов структуризации [8] или анализ произвольного текста, введенного специалистом [9], но данный подход не может гарантировать 100% соответствия правилам. Предлагается формировать текстовое описание на основании значений, введенных в соответствующие поля для каждой строки документации (каждого интервала). При этом в справочниках значения введены в именительном падеже и единственном числе, а при формировании описания необходимо получить полноценный текст на естественном языке, при этом одно и то же значение из справочника (слово) может изменяться в зависимости от остального текста, особенно это заметно на прилагательных, например «бурый», «коричневый», «буро-коричневый» и т. п.

Предлагается для каждого листа документации указать функцию, которая будет формировать описание, а в рамках данной функции с использованием всех возможностей языке JavaScript и значений текущей и предыдущей строк сформировать полноценное текстовое описание.

В данном примере «KS» — имя листа, для которого формируется описание с использованием функции getKS().

```
function getKS() {
 if(KS_FROM < KS_TO)
```

```
  KS = KS_FROM + "-" + KS_TO + " м. (" + (KS_TO - KS_FROM) + " м.).";
 else
  KS = "ОШИБКА ИНТЕРВАЛА";
 if(KS_DM)
  KS = KS + ". Диаметр бурения " + KS_DM;
 return KS;
}
```

Если нарушены условия по интервалам, то в описание будет записан текст «ОШИБКА ИНТЕРВАЛА», иначе — будут записаны параметры интервала, например «1–2 м. (1 м).», если при этом задан и диаметр бурения, то текст будет более подробным, например: «1–2 м. (1 м.). Диаметр бурения 12» (рис. 2).

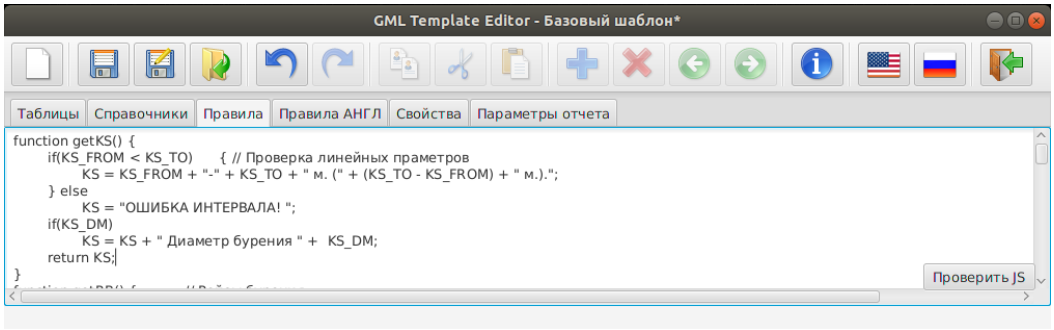

Рисунок 2. Редактирование правил создания текстового описания

Данный подход позволяет очень гибко сформировать описание. Возникает только одна особенность: при формировании описания листа «Вмещающие породы» необходимо включать текст из описаний, например, по «Минерализации», при этом привязка к конкретным строкам должна осуществляться автоматически. Особенность задачи заключается в том, что одной строке «Вмещающих пород» может соответствовать несколько строк с подынтервалами (типа «Минерализации»). В разделе 3.3 было указано, что перечень соответствующих таблиц известен заранее, соответственно, он должен быть описан в параметрах шаблона, тогда можно будет в автоматическом режиме проверить на соответствие полей между листами, но этого мало, необходимо еще сформировать общее описание по нескольким строчкам из каждого листа с подынтервалами. Это можно сделать только изменив стандартный подход к обработке функций JavaScript (в данном случае предлагается расширение возможностей языка, а не разработка нового скриптового языка).

```
//@sub.begin[MIN]
function getMIN() {
 if(!MIN_DO1_sub)
  MIN_sub = "ОШИБКА! ПОДЫНТЕВАЛ НЕ ЗАДАН. ";
 else {
```

```
      if(MIN_DO1_sub && MIN_DO2_sub)
        MIN_sub = "В подынтервалах ";
      else
        MIN_sub = "В подынтервале ";
      if(MIN_DO1_sub)
        MIN_sub = MIN_sub + "от " + MIN_OT1_sub + " м до " + MIN_DO1_sub
+ " м.";
      if(MIN_DO2_sub)
        MIN_sub = MIN_sub + ", от " + MIN_OT2_sub + " м до " +
MIN_DO2_sub + " м.";
      MIN_sub = MIN_sub + "; "
     }
     return MIN_sub;
    }
    //@sub.end
```

Для автоматической обработки все поля функции с подынтервалами заканчиваются на «_sub», что позволяет в автоматическом режиме вносить соответствующие изменения, например пронумеровав обнаруженные строки с подынтервалами числами $1, 2, ..., N$, можно таким же образом пронумеровать все переменные, что позволит «склеить» описания без конфликта по именам переменных.

Для автоматического определения параметров функции предложены две конструкции, обозначающие начало функции «//@sub.begin[MIN]» и конец функции «//@sub.end». С точки JavaScript они являются комментариями и игнорируются. При этом для поиска такой функции можно воспользоваться регулярным выражением вида

```
"\/\/\s*@sub\.begin(.+?)\/\/\s*@sub\.end"
```

При этом для получения символьного имени листа может использоваться регулярное выражение следующего вида:

```
"\[\s*(.+?)\s*\]"
```

Для получения имени функции регулярное выражение следующего вида:

```
"function\s+(.+?)\("
```

Для получения тела функции регулярное выражение следующего вида:

```
"\{(.+)\}"
```

В результате у разработчика появляется возможность автоматически несколько раз повторить код тела функции с соответствующим изменением имен переменных. Следует отметить, что описанный подход является расширением стандарта ECMAScript 5, но он представляет собой ограниченный набор конкретных правил, в целом разработка правил создания описания выполняется в соответствии со стандартом.

3.6. Формирование описания

В целом Java-код формирования описания может выглядеть следующим образом:

```
    private String evalDescription (Bindings bindings, String script,
String function) throws Exception {
     engine.setBindings(bindings, ScriptContext.ENGINE_SCOPE);
     engine.eval(script);
     Object value = ((Invocable)engine).invokeFunction(function);
     return value == null ? null : value.toString();
    }
```

При этом переменная engine является свойством соответствующего класса.

```
    private ScriptEngine engine;
    {
     ScriptEngineManager mgr = new ScriptEngineManager();
     engine = mgr.getEngineByName("nashorn");
    }
```

В данном примере переменная bindings хранит значения всех переменных, которые должны быть установлены в скрипте, переменная script содержит скрипт, требующий интерпретации, а переменная function хранит имя функции, которую необходимо вызвать.

С точки зрения использования данных из нескольких листов для формирования описания «Вмещающих пород» необходимо соответствующим образом (п. 3.5) изменять скрипт (переменная script) и в переменную bindings помещать значения не только для переменных текущего листа, но и для соответствующих листов с подынтервалами.

В данном случае становятся очевидными требования по именованию переменных в JavaScript, т. к. в одном хранилище оказываются переменные из нескольких листов (очевиден префикс в виде символического имени листа) и в рамках одного листа тоже задействуется несколько строк (очевидна необходимость в автоматическом режиме использовать постфиксы, которые не совпадут с именами переменных).

3.7. Редактирование текста

В процессе редактирования текста пользователи привыкли иметь возможность выполнять отмену и повтор последних нескольких действий. При этом используемые программы при закрытии обычно предлагают сохранить внесенные изменения. Это возможно делать, если все изменения пользователя контролировать. Наиболее удобным способом контроля в данном случае является шаблон проектирования «Фабрика» применительно к командам, который обеспечит удобный интерфейс для отслеживания каждого действия пользователя и отмены действия при необходимости.

В редакторе документации должны быть реализованы следующие команды:

– добавить строку на лист;
– удалить строку;
– вырезать строку;
– вставить строку;
– изменить значение в ячейке таблицы.

При этом необходимо учитывать, что вычисляемые поля — это такие же изменения в ячейках таблицы, которые могут быть отменены (рис. 3).

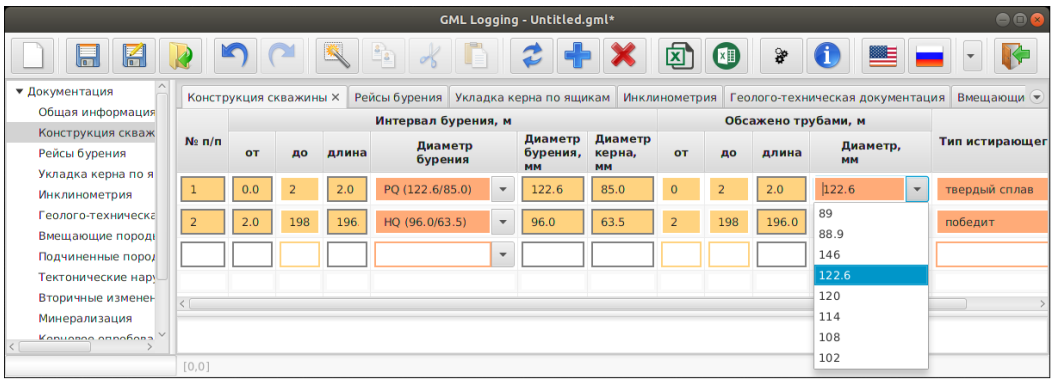

Рисунок 3. Редактирование документации

В редакторе шаблона должны быть реализованы следующие уникальные команды:

– добавление, удаление и изменение в перечне справочников;
– добавление, удаление, изменение и сортировка значений в справочниках;
– добавление, удаление, изменение значений и местоположения в таблице полей;
– копирование, вырезание и вставка значений;
– изменение текстовых полей.

При сохранении история внесенных изменений может очищаться, что позволит контролировать наличие изменений при закрытии программ.

3.8. Сохранение данных

Сохранение данных удобно делать в текстовом формате, удобном для восприятия пользователем. Таких форматов множество. Наиболее распространенные — JSON и XML. В данном случае принято решение использовать JSON. Есть множество биб-

лиотек на Java, которые позволяют работать с JSON. Исходя из простоты выбрана библиотека gson [10].

С использованием данной библиотеки создание JSON осуществляется следующими командами.

```
GsonBuilder builder = new GsonBuilder();
builder.registerTypeAdapter(MyClass.class, new MyClassSe-rializer());
Gson gson = builder.create();
String json = gson.toJson(data);
```

В данном примере MyClass — класс, требующий сериализации, MyClassServilizer — специализированный класс, обеспечивающий сериализацию. Можно было бы обойтись и без специального класса сериализации, но в исходном классе есть поля Property, которые после автоматической сериализации получаются громоздкими, читать такой JSON становится крайне не удобно. Объект data — исходный объект, подлежащий сериализации, json — результирующий объект.

Обратная операция выполняется аналогично.

```
GsonBuilder builder = new GsonBuilder();
builder.registerTypeAdapter(MyClass.class, new MyClassDe-serializer());
Gson gson = builder.create();
MyClass data = gson.fromJson(json, MyClass.class);
```

Следует отметить, что в случае большого количества данных, а количество строк в таблицах может достигать нескольких тысяч, файл становится довольно объемным. При том, что в нем хранится только текст, который может быть с легкостью упакован. Целесообразно предусматривать автоматическую упаковку и распаковку ZIP-архивов. Язык Java это поддерживает. Кроме того, в архив можно будет помещать и другие файлы, если это будет необходимо на последующих этапах развития программы.

3.9. Экспорт данных

Формат хранения данных получился специализированным, аналогичные программы работают в своих форматах. В качестве способа передачи данных в другие программы может использоваться форма Excel. Для Java существует множество библиотек работы с Excel, например Apache POI [11], которая позволяет создать необходимые листы, правильно их назвать и заполнить данными с учетом требуемого формата. Табличное хранение данных в программе позволяет создать необходимое количество листов и сохранить в Excel. Единственное, что следует выбрать, это как именовать листы и заголовки столбцов. В простейшем случае могут использоваться символьные имена, в более сложном случае экспорт может осуществляться с учетом текущего языка программы.

При необходимости с использованием Apache POI может осуществляться импорт данных в формате Excel из других программ.

3.10. Тестирование

Существует множество подходов к тестированию разработанного приложения. Большинство разработчиков используют модульные тесты, которые позволяют не только защитить поведение программы от изменения, но и упростить совместную разработку. Применение методов, описанных в [12], позволяет в автоматизированном режиме тестировать различные варианты применения программы.

4. Заключение

В результате создана небольшая по объему программа, которая позволяет решать ключевые задачи по формированию геологической документации, такой как полевой журнал. Предложен вариант решения на языке Java с использованием бесплатно распространяемых библиотек, активно используется глубокая интеграция Java и JavaScript, которая позволяет пользователю выполнять гибкую настройку формул и правил создания текстовых описаний. По сравнению с аналогами предложенное решение использует не специализированный скриптовый язык, разработанный исключительно для программы создания геологической документации, а универсальный скриптовый язык на основе стандарта ECMAScript 5, что, с одной стороны, существенно расширяет список лиц, которые могут осуществлять настройку программы, с другой стороны, за счет большого количества соответствующих учебных материалов существенно упрощает процесс освоения предложенного решения. Предполагается, что с программой предоставляется и базовый шаблон, который может адаптироваться старшим геологом под нужды конкретного месторождения. Важно, что данная адаптация не предполагает привлечения программистов, т. к. в большинстве случаев она осуществляется с использованием графического интерфейса, при этом либо добавляются и удаляются поля и листы документации, либо сокращается состав справочников базового шаблона. И только при необходимости изменения базового шаблона может потребоваться знание основ JavaScript.

С точки зрения интерфейса пользователя с учетом максимальной универсальности все листы документации представляют собой таблицы, несмотря на это программа является прямым конкурентом приложений, предлагаемых современными компаниями, в части ведения документации в полевых условиях. Использование данной программы существенно упрощает ведение полевого журнала.

Литература

[1] *Андреев В. В.* Геологическая документация : учеб. пособие. — Иркутск : Иркут. ун-т., 2000.

[2] *Солодухин М. А., Архангельский И. В., Городецкий С. И.* Новые принципы системы ведения полевой документации буровых скважин с применением современных информационных технологий // *Автоматизированные технологии изысканий и проектирования*. 2013. № 1(48), С. 66–71.

[3] *Ребрик Б. М.* Бурение инженерно-геологических скважин. Справочник. 2-е изд. перераб. и доп. — М. : Недра, 1990.

[4] *Плякин А. М.* Документация геологических наблюдений на учебных практиках: метод. указания. — Ухта : УГТУ, 2007.

[5] *Волков В. Н.* Геологическая документация и опробование поисково-разведочных выработок : учеб. пособие. — СПб., 2007.

[6] *Ананьев Ю. С.* Документация керна буровых скважин. — Томск : Изд-во Томского политехнического ун-та, 2008.

[7] Информационная система АГР 4.0 [Электронный ресурс] URL: https://agr4.ru/

[8] *Романенко С. А., Беляев С. А., Романенко Д. А.* Построение системы конкурсного отбора заявок технологической платформы // *Вестник СевКавГТИ*. 2012. Вып. 12. С. 18–21. https://elibrary.ru/item.asp?id=17734561

[9] *Belyaev S. A., Kuleshov A. S., Kholod I. I.* Solution of the Answer Formation Problem in the Question-Answering System in Russian // Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017 IEEE Conference of Russian (1–3 February 2017), SPb : LETI, 2017. P. 360–365. DOI: 10.1109/EIConRus.2017.7910567

[10] google/gson [Электронный ресурс] URL: https://github.com/google/gson

[11] Apache POI — the Java API for Microsoft Documents [Электронный ресурс] URL: https://poi.apache.org/

[12] *Черная О. С., Федорова Ю. Ю., Беляев С. А.* Применение методов и средств автоматизированного тестирования для проверки качества программных комплексов обработки измерительной информации // *Известия СПбГЭТУ «ЛЭТИ»*. 2013. Вып. 9. С. 55–58.

Автор:

Сергей Алексеевич Беляев — кандидат технических наук, доцент кафедры математического обеспечения и применения ЭВМ, Санкт-Петербургский государственный электротехнический университет (ЛЭТИ)

Development of the Geological Logging Management Program

S. A. Belyaev

Saint Petersburg Electrotechnical University "LETI"
5, Professor Popov street, Saint-Petersburg, 197376
e-mail: bserge@bk.ru

Absract. Field documentation in most cases is conducted in paper form with the use of a pencil, in rare cases licensed software is used. This is due to the high cost of equipment and the main financial investments in equipment and drilling technology, and not the means of documentation. The paper describes the construction of a universal program of field geological documentation, which solves a significant part of the tasks of its competitors in terms of field journal. The paper describes the basic requirements for the program of geological documentation, formulated its limitations, proposed solutions for each of the key tasks of development, given the fundamental fragments of the program code in Java. Approaches to the formation of a text description for each interval using the JavaScript language, including the use of data on subintervals, are described.
Keywords: field geological logging, software development, Java, JavaScript.

References

[1] *Andreev V. V.* (2000) Geologicheskaya dokumentatsiya: Ucheb. posobiye. Irkutsk, Irkut. un-t. [In Rus]
[2] *Solodukhin M. A.*, *Arkhangel'skiy I. V.*, *Gorodetskiy S. I.* (2013) *Avtomatizirovannyye tekhnologii izyskaniy i proyektirovaniya*. **1**(48):66–71. [In Rus]
[3] *Rebrik B. M.* (1990) Bureniye inzhenerno-geologicheskikh skvazhin. Spravochnik. Nedra. [In Rus]
[4] *Plyakin A. M.* (2007) Dokumentatsiya geologicheskikh nablyudeniy na uchebnykh praktikakh: metod. ukazaniya. Ukhta, UGTU. [In Rus]
[5] *Volkov V. N.* (2007) Geologicheskaya dokumentatsiya i oprobovaniye poiskovo-razvedochnykh vyrabotok. SPb. [In Rus]
[6] *Anan'yev Yu. S.* (2008) Dokumentatsiya kerna burovykh skvazhin. Tomsk. [In Rus]
[7] https://agr4.ru/
[8] *Romanenko S. A.*, *Belyayev S. A.*, *Romanenko D. A.* (2012) Vestnik SevKavGTI. 12:18–21. [In Rus]
[9] *Belyaev S. A.*, *Kuleshov A. S.*, *Kholod I. I.* (2017) Solution of the Answer Formation Problem in the Question-Answering System in Russian. In Proc. Young Researchers in Electrical and Electronic Engineering (EIConRus), 2017 IEEE Conference of Russian (1–3 February 2017), Saint Petersburg, pp. 360–365. DOI: 10.1109/EIConRus.2017.7910567
[10] https://github.com/google/gson
[11] https://poi.apache.org/
[12] *Chernaya O. S.*, *Fedorova Yu. Yu.*, *Belyaev S. A.* (2013) *Izvestiya SPbGETU «LETI».* 9:55–58. [In Rus]