УДК 004.62

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ФОРМАТОВ ОБМЕНА ДАННЫМИ, ИСПОЛЬЗУЕМЫХ В ПРИЛОЖЕНИЯХ С КЛИЕНТ-СЕРВЕРНОЙ АРХИТЕКТУРОЙ

[1]Канаев К.А., [1]Фалеева Е.В., [2]Пономарчук Ю.В.

[1]Хабаровская государственная академия экономики и права, Хабаровск, e-mail: pankrat1on@mail.ru;

[2]Дальневосточный государственный университет путей сообщения, Хабаровск, e-mail: yulia.ponomarchuk@gmail.com

Настоящая статья посвящена проведению сравнительного анализа трёх наиболее популярных форматов обмена данными, используемых разработчиками программного обеспечения в приложениях с клиент-серверной архитектурой. Представлены определения и описания рассматриваемых форматов. Также представлены критерии сравнительного анализа, которые в настоящее время представляются наиболее важными при принятии решения в процессе разработки клиент-серверного приложения. Каждый из форматов обмена данными рассмотрен с точки зрения конкретного критерия, благодаря чему удалось произвести сравнительный анализ. Сравнение трех рассматриваемых форматов обмена информацией выполнено по критериям в соответствии с данными, опубликованными не ранее 2013 года. Для количественной оценки соответствия тому или иному критерию использовалась пятибалльная шкала, как одна из самых простых и распространенных в системах оценивания. Все оценки сведены в общую таблицу. Представлены выводы на основании проведенного сравнительного анализа форматов обмена данными.

Ключевые слова: форматы обмена данными, обмен данными, клиент-серверная архитектура, сравнительный анализ, xml, json, yaml

COMPARATIVE ANALYSIS OF DATA EXCHANGE FORMATS FOR APPLICATIONS WITH CLIENT-SERVER ARCHITECTURE

[1]Kanaev K.A., [1]Faleeva E.V., [2]Ponomarchuk Y.V.

[1]Khabarovsk State Academy of Economics and Law, Khabarovsk, e-mail: pankrat1on@mail.ru;

[2]Far Eastern State Transport University, Khabarovsk, e-mail: yulia.ponomarchuk@gmail.com

The paper is dedicated to comparative analysis of the most popular data exchange formats, which are used by software developers for applications with client-server architecture. Brief descriptions of XML, JSON, and YAML are given, as well as the most important criteria for comparative analysis in order to make a decision on which format to use while develop an application. Each data exchange format is considered and some program code examples are presented. The data exchange formats are analyzed on the basis of recent literature, published since 2013. In order to do qualitative analysis 5-point scale was used as one of the simplest and commonly used. According to the results of comparative analysis JSON data exchange format is recommended to use for applications with client-server architecture.

Keywords: data exchange formats, data exchange, client-server architecture, comparative analysis, xml, json, yaml

В процессе разработки приложений с клиент-серверной архитектурой неизбежно возникает проблема выбора формата обмена данными между клиентскими и серверным модулями. Решение этой проблемы может оказать значительное влияние как на работу приложения, так и на трудоемкость дальнейшей модернизации. Время отклика, объем передаваемой информации по каналу связи, расширяемость и портируемость, требуемые ресурсы и другие параметры могут зависеть от формата обмена данными.

Целью исследования является определение критериев для проведения сравнительного анализа наиболее популярных форматов обмена данными и выбор из них наиболее отвечающего требованиям к построению новых распределенных информационных систем.

В настоящее время существует значительное количество различных форматов, рекомендуемых в литературе для использования в распределенных информационных системах. Чаще всего в сообществе разработчиков, предпочтение отдают одному из трёх наиболее используемых форматов обмена данными: XML, JSON, YAML.

XML (Extensible Markup Language) – простой, очень гибкий текстовый формат, являющийся подмножеством SGML (ISO 8879) [3], который позволяет определять собственные теги и атрибуты. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать ее в соответствии с особенностями конкретной предметной области, будучи ограниченным лишь синтаксическими правилами языка [4].

Возможность создания собственных тегов делает XML универсальным.

JSON (Java Script Object Notation) – представляет собой облегченный формат обмена данными между компьютерами [5]. В соответствии с определением стандарта сценарного языка программирования ECMA (Европейской ассоциации производителей компьютеров), он является производным от литералов Java Script. JSON более компактен, чем XML, его конструкции легче анализируются средствами Java Script, для которого JSON является внутренним используемым типом данных. Основная сфера применения JSON – программирование web-приложений, где он служит альтернативой XML.

YAML – человекочитаемый формат сериализации данных, концептуально близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования. В настоящее время акроним YAML интерпретируется как «YAML Ain't Markup Language» («YAML – не язык разметки»). В названии отражена история развития: на ранних этапах акроним представлял собой аббревиатуру выражения «Yet Another Markup Language» («Ещё один язык разметки») и даже рассматривался как конкурент XML, но позже был переименован, чтобв акцентировать внимание на данных, а не на разметке документов.

В настоящее время не существует сформулированных критериев, благодаря которым можно было бы сравнить форматы обмена данными в приложениях с клиент-серверной архитектурой. Однако на различных ресурсах, посвященных IT-тематике, профессиональными разработчиками и архитекторами информационных систем не раз делались попытки такие критерии рекомендовать. Стоит отметить, что, несмотря на значительное число характерных черт, рекомендованных в качестве критериев сравнения, не все они представляют теоретическую или практическую пользу. Ниже представлены те критерии сравнительного анализа, которые в настоящее время представляются наиболее важными при принятии решения в процессе разработки клиент-серверного приложения.

● Человекочитаемость – предполагает простую и удобную разметку передаваемых данных. При этом язык должен иметь незначительное количество символов-разделителей (скобки, кавычки и т.д.).

● Простота сериализации – преобразования объекта (данных) в поток байтов для дальнейшего хранения или передачи по каналу связи, в память или файл.

● Простота десериализации – преобразования потока байтов в объект данных.

● Возможность проверки формата входных данных – наличие в формате обмена данными внутреннего языка описания структуры документа (JSON-Schema, XML-Schema), необходимого для осуществления предварительной проверки на соответствие приходящих данных, например, со стороны клиента.

● Эффективность сжатия данных – включает скорость выполнения алгоритма компрессии и коэффициент сжатия.

● Распространенность – наличие большого количества разработчиков, использующих тот или иной формат обмена данными.

● Динамика развития, которая характеризуется скоростью популяризации и развития.

Сравнение трех рассматриваемых форматов обмена информацией будет выполнено по вышеперечисленным критериям в соответствии с данными, опубликованными не ранее 2013 года. Для количественной оценки соответствия тому или иному критерию используется пятибалльная шкала как одна из самых простых и распространенных в системах оценивания. Затем подсчитывается средний балл оценки каждого формата. Следует отметить, что используемый метод сравнительного анализа является субъективным, так как проставленные оценки являются количественной характеристикой выводов и опыта авторов настоящей статьи.

Удобство чтения формата для обмена данными вызывает споры среди разработчиков программного обеспечения, т.к., по мнению многих, является слишком субъективным. Некоторые специалисты утверждают, что это один из самых важных критериев, и приводят доказательства того, что один формат удобнее для чтения, нежели другой. Так, в результате даже беглого просмотра интернет-ресурсов, можно сделать вывод о неудобстве (более трудном восприятии) XML.

Для рассмотрения второй характеристики необходимо привести примеры преобразования объекта данных (XML, JSON и YAML) в серверном модуле информационной системы. В качестве языка программирования для иллюстрации выбран C# как один из самых популярных языков с большим количеством дополнительных библиотек и поддержкой миллионов разработчиков программного обеспечения.

Пример сериализации объекта XML:

```
XmlSerializer x = new XmlSerializer(typeof(Students));

TextWriter writer = new StreamWriter(filename);

x.Serialize(writer, Emps);
```

Пример сериализации объекта JSON:

```
List<Student> students = new List<Student>();

JavaScriptSerializerserializer = JavaScriptSerializer();

string json = serializer.Serialize(students);
```

Пример сериализации объекта YAML:

```
List<Student> students = new List<Student>();

varserializer = new YamlSerializer();

string yaml = serializer.Serialize(students);
```

По приведенным примерам можно заключить, что в целом все три формата достаточно удобны и просты для сериализации объектов данных. Заметим, что JSON и YAML имеют одинаковую структуру программного кода сериализации объекта, отличаясь лишь наименованием собственных типов. Два указанных формата похожи друг на друга и отличаются лишь синтаксическими разделителями.

Ниже приведены примеры десериализации объекта данных для пояснения принятия решения по третьему критерию.

Пример десериализации объекта XML:

```
XmlSerializerx = new XmlSerializer(typeof(Students));

StreamReader reader = new StreamReader(path);

student = (Students)x.Deserialize(reader);
```

Пример десериализации объекта JSON:

```
 JavaScriptSerializerserializer = JavaScriptSerializer();

 Students student = new Students();

student = serializer.Deserialize<Students>(strJson);
```

Пример десериализации объекта YAML:

```
varserializer = new YamlSerializer();

Students student = new Students();

student = serializer.Deserialize<Students>(strJson);
```

В этом случае также можно заключить, что все три формата являются удобными и достаточно простыми.

Наличие возможности проверки формата входных данных должно рассматриваться как для серверной, так и для клиентской частей информационной системы. При использовании XML и JSON есть возможность осуществить проверку данных через язык описания структуры документа – schema. В YAML такая операция уже внедряется, однако пока не получила широкого распространения и поддержки среди разработчиков. Однако в случае клиентского модуля, как правило, сложнее реализовать обнаружение ошибок в данных всех трех форматов.

Реализация алгоритмов сжатия данных не является распространенной задачей при передаче данных в рассматриваемых форматах. Однако в некоторых приложениях компрессия данных действительно необходима, например в сетях мобильных устройств и подвижных объектов с целью экономии энергии и уменьшения трафика. Немногие компании и разработчики проводили исследования и тестирование с целью сравнительного анализа. По результатам тестирования бенчмарка, JSON оказался более компактным, а также показал лучшую эффективность сжатия по сравнению с XML.

При использовании более сложных и нестандартных алгоритмов, увеличивающих

трудоёмкость процесса разработки, можно добиться хороших показателей и у XML. Некоторые из эффективных методов сжатия данных в XML формате описаны в технических материалах на сайте IBM [2]. Следует отметить отсутствие опубликованных результатов тестирования алгоритмов сжатия для формата YAML. Можно предположить, что у исследователей пока не возникло интереса к данному вопросу в связи со специфической областью применения этого формата.

Распространенность и динамика развития форматов обмена данными не менее важны для принятия решения о дальнейшем их использовании в распределенных информационных системах, поскольку от популярности какой-либо технологии зависит её поддержка сообществом разработчиков программного обеспечения и развитие в будущем. Результаты проведенного исследования, проводимого в конце 2013 года, показали значительно растущую популярность JSON в сравнении с XML. По YAML информация отсутствует.

Результаты сравнительного анализа форматов обмена данными для приложений с клиент-серверной архитектурой: XML, JSON, YAML – приведены в таблице.

Сравнительный анализ форматов обмена данными

Критерий	XML	JSON	YAML
Удобство чтения (человекочитаемость)	4	5	4
Простота сериализации	5	5	5
Простота десериализации	5	5	5
Возможность проверки входных данных	4	4	3
Эффективность сжатия данных	3	5	1
Распространенность	4	5	1
Динамика развития	3	5	1
Средний балл	4	4,9	3

По результатам проведенного исследования, отраженным в таблице, можно сделать вывод о том, что, несмотря на значительное количество web-ресурсов и сервисов, документации и библиотек, применяющих в качестве формата обмена данными XML, при построении новых распределенных информационных систем следует рассмотреть использование JSON. В настоящее время в профессиональном сообществе разработчиков программного обеспечения достаточно часто отмечают то, что этот формат является быстро развивающимся и простым для реализации обмена данными в системах. Заметим, что формат обмена данными YAML был предложен сравнительно недавно и только начинает свой путь развития, но уже сейчас составляет серьезную конкуренцию аналогам по некоторым характеристикам. Основным преимуществом YAML, по заверению его создателей, является удобство чтения. Однако утверждение достаточно спорно, учитывая наличие большого количества возможностей других форматов и опыт работы с ними.

В дальнейшем предполагается использование полученной методики критериев при разработке различных распределенных информационных систем. Представленные критерии помогут определиться разработчикам программного обеспечения в необходимости внедрения какого-либо формата обмена данными. Так как все представленные форматы достаточно активно развиваются, то предлагается использование не результатов сравнительного анализа, а именно использование критериев для определения наилучшего формата на текущий период.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 13-07-00615.

Список литературы

1. Э. Таненбаум, М. ван Стеен. Распределенные системы принципы и парадигмы. – СПб.: Питер, 2003.

2. Д. Мерц, XML Matters: Вопросы XML: XML и сжатие. Электронный ресурс. IBM developerWorks Россия: Ресурс IBM для разработчиков и IT профессионалов. URL: http://www.ibm.com/developerworks/ru.

3. Extensible Markup Language (XML). – URL: http://www.w3.org/XML/ (дата обращения 02.03.2015).

4. Вугт В.В. OpenXML Кратко и доступно. – Изв-во: Microsoft Press, 2007. – 109с.

5. Ben Smith. Beginning JSON. – Apress, 2015. – 295 p.

References

1. Je. Tanenbaum, M. van Steen. Raspredelennye sistemy principy i paradigmy. SPb.: Piter, 2003.

2. D. Merc, XML Matters: Voprosy XML: XML i szhatie. Jelektronnyj resurs. IBM developerWorks Rossija: Resurs IBM dlja razrabotchikov i IT professionalov. URL: http://www.ibm.com/developerworks/ru.

3. Extensible Markup Language (XML). URL: http://www.w3.org/XML/ (data obrashhenija 02.03.2015).

4. Vugt V.V. OpenXML Kratko i dostupno. Izv-vo: Microsoft Press, 2007. 109s.

5. Ven Smith. Beginning JSON. Apress, 2015. 295 p.

Рецензенты:

Графский О.А., д.т.н., доцент, профессор кафедры «Вычислительная техника и компьютерная графика», Дальневосточный государственный университет путей сообщения, г. Хабаровск;

Криштоп В.В., д.ф.-м.н., профессор, проректор по учебной работе, Дальневосточный государственный университет путей сообщения, г. Хабаровск.