УДК 004.045, 004.773

А. М. Романченко

ОБОБЩЕННАЯ СТРУКТУРНАЯ МЕТАМОДЕЛЬ ПРОТОКОЛА ИНФОРМАЦИОННОГО ВЗАИМОДЕЙСТВИЯ

Романченко А.М. **Обобщенная структурная метамодель протокола информационного взаимодействия.**

Аннотация. Данная работа посвящена разработке структурной метамодели произвольного протокола информационного взаимодействия. Эта метамодель может быть использована для создания формализованных моделей различных протоколов, построения автоматических декодеров, хранения и обмена информацией о протоколах. Ее использование позволяет упростить многие прикладные задачи анализа данных, сравнения различных протоколов, влияния ошибок в канале связи на правильность декодирования и т.д.

Ключевые слова: построение протоколов, моделирование протоколов, анализ протокольных данных, протоколы прикладного уровня.

Romanchenko A.M. **Generalized Structural Metamodel of Information Interaction Protocol.**

Abstract. This work is devoted to the development of structural meta-model arbitrary protocol information interaction. This meta-model can be used to create formal models of various protocols, construction of automatic decoders by model, storage and exchange of information on protocols. Its use simplifies many application tasks of data analysis, tasks of comparing various protocols, study effect of errors in the communication channel on correctness of decoding.

Keywords: construction of protocols, modeling of protocols, analysis of protocol data, application protocols.

1. Введение. Сегодня, несмотря на достаточно большое количество различных протоколов информационного взаимодействия, используемых при реализации большого спектра информационных технологий, постоянно возникает необходимость разработки новых протоколов, особенно прикладного уровня. Разработка нового протокола автоматически означает необходимость решения задач распознавания, декодирования, проверки целостности и анализа протокольных данных. В данной работе предлагается использовать обобщенную метамодель протокола информационного взаимодействия в качестве основы для разработки новых протоколов, их описания в виде некоторой реализации этой модели, использовании для обмена информацией, построения декодеров, распознавания и анализа.

Преимуществом использования обобщенной метамодели является возможность описания в рамках этой модели произвольного протокола и использования его в дальнейшем для решения конкретных прикладных и аналитических задач. При этом, если реализовать построение декодера на основе информации содержащейся в обобщенной метамодели, то добавление нового протокола в

программную реализацию или изменение существующего не потребует изменения декодера, достаточным будет добавления новой модели протокола. Также удобным является возможность ведения базы данных известных протоколов и построение на основе обобщенной метамодели математических и других моделей существующих и новых протоколов.

Обобщенная структурная метамодель протоколов, предположительно, не подходит для описания слабо структурированных протоколов высокой сложности, в которых могут присутствовать, или не присутствовать множество полей переменной длины, в том числе с вложенной информацией, подлежащей самостоятельному интерпретированию. К таким протоколам можно отнести HTTP, FTP, POP3 и другие аналогичной сложности. Но для таких протоколов и не требуется использовать обобщенный декодер или анализатор, так как для них существует множество стандартных библиотек обработки.

При проведении анализа существующих подходов к разработке и анализу протоколов не было обнаружено специализированных инструментов их проектирования и стандартного описания структуры протокола в формализованном виде. На практике почти все разработчики используют спецификацию протокола, то есть его текстовое описание. Очевидно, что информация о протоколе, представленная в форме текстового описания, не может быть использована для решения задач, возникающих на практике (подробно рассмотрены далее), так как невозможна ее автоматическая интерпретация.

Ранее в ряде работ [3, 4] уже предлагалось использование модели протокола на языке XML вместо спецификации. Основная идея, предложенная в этих работах, состоит в описании с помощью языка XML последовательности действий, составляющих протокол, и связи их с функциями программы, реализующей эти действия. Это, в основном, относится к многошаговым протоколам (например криптографическим), в которых взаимодействуют несколько пользователей. Описание данных протокола, предложенное в работе [4], содержит информацию только об их составе и недостаточно для их обобщенной обработки. Протоколы, описываемые в работе [3] (и другие протоколы семейства XMLP) фактически ограничены только текстовой формой, что не всегда эффективно на практике и существенно ограничивает разработчика. То есть XML в работе [3] используется не для описания формата протокола, а для представления данных протокола. Ни один из рассмотренных подходов, в отличие от предложенного в работе, не позволяет описать

структуру произвольного протокола, и построить единый декодер для множества моделей протоколов.

Ряд других работ по данной тематике предлагают описание протокола с помощью специализированных языков, таких как Ciecero [9], Promela++[5], LOTOS [7,8], Estelle [6] и других. Использование таких моделей потребует от разработчиков знания специфичного языка и связи его с основным языком разработки (C++, java, C#), поэтому применение такой модели для решения практических задач будет затруднительным, если вообще возможным. Язык XML, использованный для построения метамодели в данной работе, свободен от этих недостатков и легко интегрируется в программу на любом из указанных языков программирования. Язык XML достаточно прост для понимания, поэтому и начинающие специалисты и даже пользователи будут способны самостоятельно разработать протокол для своих нужд после освоения подхода предложенного в работе.

2. Разработка метамодели протокола с использованием языка XML/DTD. При проведении анализа подмножества существующих протоколов информационного обмена было выявлено, что все хорошо структурированные протоколы могут быть классифицированы по признаку вида содержащихся в них данных. На его основе можно выделить:

1. Информационные протоколы — протоколы, содержащие поля определенной длины, в которых находятся заранее известные данные, такими, например, являются многие телеметрические протоколы. Обычно поля являются числами различных форматов, но могут содержать и бинарные данные.

2. Транспортные протоколы — протоколы, предназначенные для «упаковки» произвольных данных. Обычно это требуется для передачи данных с верхних уровней эталонной модели взаимодействия открытых систем (ЭМВОС) на нижние уровни. Упаковка данных часто используется при адаптации существующих протоколов к передаче с помощью специфической аппаратуры, а также сопряжении разнородных сетей. Обычно в рамках транспортного протокола неизвестно, какая именно информация передается с ее помощью. Поля, в которые осуществляется упаковка данных, могут быть бинарными или текстовыми (например, в кодировке BASE64), постоянной или переменной длины.

3. Смешанные протоколы, в которых присутствуют как числовые поля с известным содержанием и правилами его интерпретации, так и поля с произвольными данными заранее неизвестной структуры.

4. Протоколы являющиеся «системой команд» некоторого устройства или программы.

Целью разработки обобщенной структурной метамодели протокола является предоставление возможности описания моделей всех приведенных видов протоколов. При этом информация, содержащаяся в модели, должна быть достаточной для построения декодера без модификации программы обработки для каждого протокола. Модель должна быть интуитивно понятной пользователю и доступной для генерации, интерпретации и изучения без использования специальных средств.

В качестве инструмента построения обобщенной модели протокола информационного взаимодействия, будет использован язык XML с использованием механизма DTD (data type definition). Выбор такого инструментария позволит достичь сразу нескольких основополагающих целей работы:

1. Стандартизацию, так как язык XML является широко распространенным и используется для описания моделей данных в рамках некоторой предметной области, формализованной с помощью этого же языка.

2. Возможность создания протоколов, их изучения и использования с помощью общедоступных инструментов, в том числе и общего назначения, например, любого текстового редактора. При этом все же рекомендуется воспользоваться специализированными инструментами для работы с файлами формата XML, способными автоматически проверять синтаксис файла и соответствие его файлу определения данных. Автором был использован в качестве такого инструмента «XML copy editor» - http://xml-copy-editor.sourceforge.net/.

3. Возможность автоматической проверки корректности описания протокола в рамках выбранной модели данных. Существование большого количества интерпретаторов формата XML, позволяющих извлекать информацию из модели конкретного протокола, дает возможность построения «на лету» декодера по конкретной реализации протокола с помощью обобщенной модели. Такие инструменты существуют для большинства распространенных языков программирования, в частности для C++ можно назвать:

- Xerces (http://xerces.apache.org/);
- Arabica (http://www.jezuk.co.uk/cgi-bin/view/arabica);
- TinyXML (http://sourceforge.net/projects/tinyxml/);
- pugixml (https://code.google.com/p/pugixml/);
- libxml++ (http://libxmlplusplus.sourceforge.net/).

Для построения обобщенной структурной метамодели протокола необходимо выяснить, из каких элементов может состоять протокол информационного обмена, какие эти элементы имеют характеристики, а также какие информационные поля имеет сам протокол.

Все структурные элементы, входящие в состав произвольного протокола, можно разделить на два вида — содержательные и служебные. Содержательные элементы предназначены для передачи информации с помощью протокола, служебные необходимы для функционирования самого протокола и использующей его инфраструктуры — программных и аппаратных средств. В качестве модели представления протокольных данных будем рассматривать поток байтов, представляющих протокольные блоки с возможными искажениями разных видов (инверсия, вставка/пропуск и т.п.) вследствие ошибок в канале связи. К служебным элементам произвольного протокола информационного обмена можно отнести заголовок (префикс), окончание (постфикс) и контрольную сумму. К содержательным полям протоколов можно отнести числовые поля разных форматов, текстовые или бинарные поля постоянной и переменной длины, поля флагов и т. п. Метамодель должна позволять описывать все эти элементы с детализацией, достаточной для построения декодера. Упрощенная структура метамодели в графическом виде приведена на рисунке 1. Рассмотрим элементы, представленные в ней и их характеристики более детально.

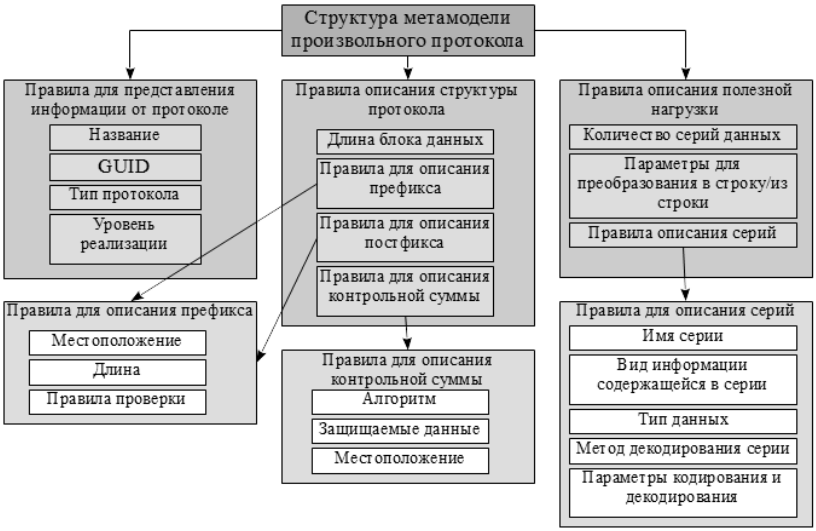

Рис. 1. Упрощенная структура метамодели протокола

Информационные характеристики протокола могут быть представлены следующим набором параметров:

– тип протокола, числовое поле, может принимать значения из множества: «информационный», «транспортный», «система команд», «смешанный» (информационно-транспортный), «другой»;

– уровень реализации протокола в рамках ЭМВОС, числовое поле, может принимать значения из множества: «физический/1», «канальный/2», «сетевой/3», «транспортный/4», «сеансовый/5», «представительский/6», «прикладной/7» или «неизвестный» если место функционирования протокола неизвестно или не определено;

– название протокола, текстовое поле переменной длины;

– глобальный уникальный идентификатор GUID — текстовая строка формата «XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX» где «X» - шестнадцатеричная цифра, использование GUID практически гарантирует уникальность различных протоколов даже при использовании большого их количества;

– длина блока данных, может быть константой или лежать в некотором допустимом диапазоне;

– число элементов (серий) данных, каждая серия содержит один из содержательных элементов полезной нагрузки, например числовое поле (время, температуру) или бинарное поле — произвольные данные, возможно, имеющие собственную сложную структуру.

Префикс может входить или не входить в состав протокола, как правило, его присутствие облегчает декодирование, но уменьшает объем полезной нагрузки, передаваемой протоколом. Можно выделить следующие содержательные характеристики описывающие префикс:

– наличие префикса;

– смещение от начала блока данных (обычно равно 0, то есть префикс является началом блока данных);

– длина (обычно используются длины от 1 до 4 байт);

– значение префикса.

Проверка префикса выполняется путем простого сравнения с эталоном, поэтому такое понятие как алгоритм проверки для этого элемента представляется излишним.

Постфикс имеет такие же характеристики, как и префикс, но используется для обозначения конца блока данных.

Серия данных (элемент полезной нагрузки) является наиболее важной частью любого протокола, определяет, что и как передается в его составе. Название «Серия» использовано потому, что набор

значений одного параметра в последовательных протокольных блоках данных образует серию данных, например серию отсчетов времени. Важнейшими характеристиками серии данных являются:

– имя серии, текстовое поле переменной длины;

– тип значения, четырехсимвольный код, подробно рассмотрен ниже;

– тип данных в серии, четырехсимвольный код, подробно рассмотрен ниже;

– параметры серии, определяющие длину поля данных, необходимы только для некоторых типов серий;

– тип декодера для серии данных.

При изучении существующих протоколов, были выделены ряд типов использованных в них серий, которые представлены на рисунке 2. Всего в нем отражено 17 основных типов, но при необходимости этот список может быть расширен, например 64 битными типами данных.

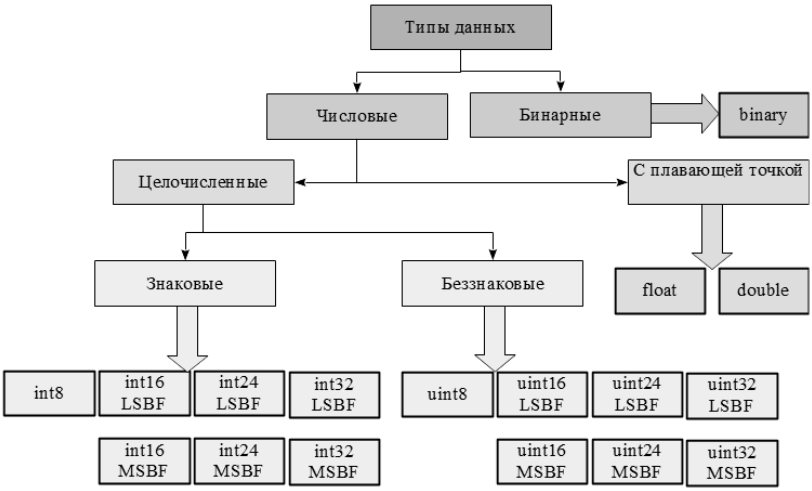

Рис. 2. Типы серий данных, использованных в метамодели

С точки зрения практической реализации для кодирования типа данных удобно использовать т. н. «четырехсимвольный код» или FourCC(four characters code). С одной стороны его использование достаточно наглядно при программной реализации протокола, особенно с использованием специального макроса. С другой стороны использование 4-байтных значений кода обеспечивает максимальную производительность и удобство манипуляции на большинстве

аппаратных платформ. Пример использования такого кодирования приведен в листинге 1 (язык C++).

```
#define DW_Make(btL, btM1, btM2, btH)
((static_cast<DWORD>(btH)<<24)|(static_cast<DWORD>(btM2)<<16)|
(static_cast<DWORD>(btM1)<<8)| (static_cast<DWORD>(btL)))

// пример кодирования типа int32 LSBF
static DWORD GetSID(void){return DW_Make(' ','L','I','4');}
```

Листинг 1. Пример использования FourCC для кодирования типа серии

Но использование кодов FourCC непосредственно в модели протокола неудобно из-за сложности запоминания четырехсимвольных аббревиатур, поэтому предлагается использовать специальный элемент для связи четырехсимвольного типа и его расширенного названия, удобного для использования в модели. Пример такой связи с помощью тега «ASSOC» представлен в листинге 2.

```
<ASSOC Type="item" From="int32LSBF" To=" LI4"/>
```

Листинг 2. Связь кода FourCC с интуитивно понятным названием типа данных, использованным при построении модели

Атрибут «From» обозначает название, использованное в модели, атрибут «To» - кодирование в программной реализации, обязательно должен иметь длину 4 символа. Атрибут «Type» означает, для какого элемента используется отображение, то есть данный механизм отображения можно использоваться не только для удобного кодирования типа данных, но и для других элементов, например типа серии.

Тип данных в серии предназначен для отражения того, какая именно информация в ней передаются. Например, температуру, считанную с первого датчика, можно сопроводить кодом типа данных, представленным в листинге 3. Использование такого типа кодирования имеет несколько важных преимуществ, например, дает возможность легко отображать структуру протокола в виде дерева, содержащего информацию о составе его полей, а также автоматически строить графические модели протокола.

```
// кодирование в модели
<SeriesType>' ','T','*','1'</SeriesType>

// кодирование в программе обработки
#define ADC_T1 DW_Make(' ','T','*','1')
```

Листинг 3. Пример кодирования типа информации в серии

Но самым важным преимуществом такого кодирования является возможность нахождения и специализированной обработки данных

нужных типов в составе различных (произвольных) протоколов. Это позволяет использовать унифицированные алгоритмы обработки любых протоколов, содержащих в своем составе серии нужных типов.

Характеристика «*тип декодера*» должна содержать всю необходимую информацию для правильного извлечения данной серии из пакета. Предлагается использование следующих видов декодеров:

– фиксированное смещение, для серий постоянной известной длины;

– фиксированное смещение и размер, для серий различной длины, например бинарных полей;

– относительное смещение от начала или конца пакета;

– фиксированное смещение, значение которого записано в другой серии.

На практике может потребоваться больше типов декодеров, поэтому приведенный список предлагается рассматривать в качестве основы для дальнейшего расширения по мере необходимости.

Контрольная сумма — важнейшая характеристика протокола позволяющая проверять его целостность, также в простых протоколах (причем фиксированной длины) может быть использована в качестве механизма синхронизации границы пакета. Имеет следующие характеристики:

– смещение начала данных от начала пакета;

– смещение конца данных от конца пакета;

– длина контрольной суммы в байтах;

– тип алгоритма;

– смещение, по которому располагается контрольная сумма от начала или конца пакета.

Важно понимать, что смещение начала данных от начала пакета, а конца данных от конца пакета, позволяет удобным образом работать с пакетами переменной длины.

Дополнительно к приведенным характеристикам предлагается использование в описании протокола механизма конвертации в строку, чтобы иметь возможность сохранять/загружать данные произвольного протокола в строковой форме. Предполагается, что каждый пакет будет соответствовать одной строке в сохраненных данных. Поля разделяются пробелом, бинарные/текстовые поля преобразуются в кодировку BASE64 или аналогичные. Характеристики механизма преобразования определяют, использовать ли нумерацию строк при сохранении и порядок сохранения данных, который может отличаться от порядка следования данных в пакете. Кроме правила конвертации предлагается использование правила проверки соответствия некоторой

строки протоколу, это правило должно определять алгоритм проверки и задавать параметры этого алгоритма. Предлагается использование следующих алгоритмов проверки корректности:

– подсчет количества разделителей (пробелов);
– проверка числа разделителей и допустимых символов;
– использование регулярного выражения, это наиболее прогрессивный способ проверки, пример такого выражения для протокола содержащего только числовые поля: «^(?>[+-]?\d+(?>\.\d+(?>e[+-]\d+)?)?(?>\s+)?){N}$», где N – это число серий данных увеличенное на 1.

Метамодель произвольного протокола в форме файла data type definition представлена в листинге 4.

```
<!ELEMENT protocols (ASSOC+, protocol)>
<!ELEMENT ASSOC EMPTY>
<!ATTLIST ASSOC Type (item) #REQUIRED>
<!ATTLIST ASSOC From CDATA #REQUIRED>
<!ATTLIST ASSOC To CDATA #REQUIRED>
<!ELEMENT protocol (Title, Prefix, Postfix, GUID, Length, SeriesCount, Series*, CRCType, FormatString, CheckString?)>
<!ATTLIST protocol type (DATA|TRANSPORT|COMMANDSYSTEM|MIX|OTHER) #REQUIRED>
<!ATTLIST protocol level (UNDEFINED|L1 |PHYSICAL|L2 |CHANNEL|L3 |NETWORK|L4 |TRANSPORT|L5 |SESSION|L6 |PRESENTATION|L7 |APPLICATION) #REQUIRED>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Prefix EMPTY>
<!ATTLIST Prefix present (YES|NO) #REQUIRED>
<!ATTLIST Prefix offset CDATA #REQUIRED>
<!ATTLIST Prefix type (None|Simple1B|Simple2B|Simple3B|Simple4B) #REQUIRED>
<!ATTLIST Prefix value CDATA #REQUIRED>
<!ELEMENT Postfix EMPTY>
<!ATTLIST Postfix present (YES|NO) #REQUIRED>
<!ATTLIST Postfix offset CDATA #REQUIRED>
<!ATTLIST Postfix type (None|Simple1B|Simple2B|Simple3B|Simple4B) #REQUIRED>
<!ATTLIST Postfix value CDATA #REQUIRED>
<!ELEMENT GUID (#PCDATA)>
<!ELEMENT Length EMPTY>
<!ATTLIST Length min CDATA #REQUIRED>
<!ATTLIST Length max CDATA #REQUIRED>
<!ELEMENT SeriesCount (#PCDATA)>
<!ELEMENT Series (SeriesName, SeriesType, SeriesParam?, DecodeType)>
```

```
<!ATTLIST Series VarType (int8| int16LSBF| int24LSBF| int32LSBF| int16MSBF|
int24MSBF| int32MSBF| uint8| uint16LSBF| uint24LSBF| uint32LSBF| uint16MSBF|
uint24MSBF| uint32MSBF| FLOAT| DOUBLE| Binary) #REQUIRED>
<!ELEMENT SeriesName (#PCDATA)>
<!ELEMENT SeriesType (#PCDATA)>
<!ELEMENT SeriesParam EMPTY>
<!ATTLIST SeriesParam Length CDATA #IMPLIED>
<!ATTLIST SeriesParam MinLength CDATA #IMPLIED>
<!ATTLIST SeriesParam MaxLength CDATA #IMPLIED>
<!ATTLIST SeriesParam DefaultLength CDATA #IMPLIED>
<!ELEMENT DecodeType (FIXEDOFFSET| FIXED| RELATIVEBE|
FIXOFF_SERIESLENGTH)>
<!ELEMENT FIXEDOFFSET EMPTY>
<!ATTLIST FIXEDOFFSET Offset CDATA #REQUIRED>
<!ELEMENT FIXED EMPTY>
<!ATTLIST FIXED Offset CDATA #REQUIRED>
<!ATTLIST FIXED Length CDATA #REQUIRED>
<!ELEMENT RELATIVEBE EMPTY>
<!ATTLIST RELATIVEBE OffsetFromBegin CDATA #REQUIRED>
<!ATTLIST RELATIVEBE OffsetFromEnd CDATA #REQUIRED>
<!ELEMENT FIXOFF_SERIESLENGTH EMPTY>
<!ATTLIST FIXOFF_SERIESLENGTH Offset CDATA #REQUIRED>
<!ATTLIST FIXOFF_SERIESLENGTH LengthIndex CDATA #REQUIRED>
<!ELEMENT CRCType (#PCDATA)>
<!ATTLIST CRCType databegin CDATA #REQUIRED
dataend CDATA #REQUIRED lengthvalue CDATA #REQUIRED
offsetvalue CDATA #REQUIRED
algtype (XOR|CRC8|ADD8|None|MD5) #REQUIRED>
<!ELEMENT FormatString (OrderData?)>
<!ATTLIST FormatString UseLineNumber (YES|NO) #REQUIRED>
<!ELEMENT OrderData EMPTY>
<!ATTLIST OrderData remap CDATA #REQUIRED>
<!ELEMENT CheckString EMPTY>
<!ATTLIST CheckString type (None| SpaceCount| SpaceAndExclude| RegEXP)
#REQUIRED>
<!ATTLIST CheckString param1 CDATA #REQUIRED>
<!ATTLIST CheckString param2 CDATA #REQUIRED>
```

Листинг 4. Метамодель произвольного протокола в форме файла DTD

В качестве иллюстрации применения данной метамодели для построения протоколов разных типов приводится два примера моделей информационного и транспортного протокола.

Простой информационный протокол содержит три числовых серии данных — три значения угловой скорости по ортогональным осям X, Y, Z, может быть отнесен к телеметрическим протоколам. Он имеет длину пакета 14 байт, префикс и контрольную сумму формата

CRC8. Данные в нем представлены тремя полями с плавающей точкой формата «float» (размерности 4 байта). Модель этого протокола приведена в листинге 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE protocols SYSTEM "Protocols.dtd">
<protocols>
<ASSOC Type="item" From="FLOAT" To=" F4"/>
<protocol type="DATA" level="APPLICATION">
<Title>3 GYRO</Title>
<Prefix present="YES" offset="0" type="Simple1B" value="0xec"/>
<Postfix present="NO" offset="0" type="None" value="0x00"/>
<GUID>7FAC6981-4F66-4728-B023-C9BE99529BC9</GUID>
<Length min="14" max="14" />
<SeriesCount>3</SeriesCount>
<Series VarType="FLOAT">
<SeriesName>Гир. X</SeriesName>
<SeriesType>' ',' ','W','X'</SeriesType>
<DecodeType> <FIXEDOFFSET Offset="1" /> </DecodeType>
</Series>
<Series VarType="FLOAT">
<SeriesName>Гир. Y</SeriesName>
<SeriesType>' ',' ','W','Y'</SeriesType>
<DecodeType> <FIXEDOFFSET Offset="5" /> </DecodeType>
</Series>
<Series VarType="FLOAT">
<SeriesName>Гир. Z</SeriesName>
<SeriesType>' ',' ','W','Z'</SeriesType>
<DecodeType> <FIXEDOFFSET Offset="9" /> </DecodeType>
</Series>
<CRCType databegin="0" dataend="12" lengthvalue="1" offsetvalue="13"
algtype="CRC8"/>
<FormatString UseLineNumber="YES" />
<CheckString type="None" param1="" param2=""/>
</protocol>
</protocols>
```

Листинг 5. Пример модели простого информационного протокола

Второй протокол относится к классу транспортных протоколов, то есть инкапсулирует произвольные данные протоколов других уровней. Он содержит одно бинарное поле длины 64 байта, защищенное контрольной суммой MD5 длиной 16 байт. Общая длина пакета составляет 80 байт. Модель данного протокола приведена в листинге 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE protocols SYSTEM "Protocols.dtd">
<protocols>
<ASSOC Type="item" From="Binary" To=" BIN"/>
<protocol type="DATA" level="APPLICATION">
<Title>Transport simple</Title>
<Prefix present="NO" offset="0" type="None" value="0x00"/>
<Postfix present="NO" offset="0" type="None" value="0x00"/>
<GUID>3738BE36-107D-4C74-80F9-086C9924100F</GUID>
<Length min="80" max="80" />
<SeriesCount>1</SeriesCount>
<Series VarType="Binary">
<SeriesName>Payload</SeriesName>
<SeriesType>' ','b','i','n'</SeriesType>
<SeriesParam Length="64" />
<DecodeType> <FIXED Offset="0" Length="64" /> </DecodeType>
</Series>
<CRCType databegin="0" dataend="63" lengthvalue="16" offsetvalue="64"
algtype="MD5"/>
<FormatString UseLineNumber="NO" />
<CheckString type="None" param1="0" param2="0" />
</protocol>
</protocols>
```

Листинг 6. Пример модели простого транспортного протокола

Для сравнения рассмотрим пример описания данных протокола из работы [4] см. листинг 7.

```
<object>
        <name>Session_State</name>
        <field type=”String”>id</field>
        <field type=”int”>Na</field>
        <field type=”int”>Nb</field>
        <field type=”key”>PrivateKey</field>
<field type=”key”>PublicKey</field>
</object>
```

Листинг 7. Пример описания данных предназначенных для использования высокоуровневых функций обработки

Очевидно, что использовать такую модель для обобщенного декодирования и других видов обработки невозможно из-за недостаточной информации о свойствах передаваемых данных, поэтому такая модель подходит только для использования частных обработчиков, которым известен не только состав, но и структура данных протокола.

3. Использование метамодели протокола при решении прикладных задач. Рассмотрим, как предложенная метамодель и модели протоколов, построенные с ее использованием, могут быть применены на практике.

Обмен информацией о протоколах — так как, по сути, предложенная модель представляет собой некий стандарт описания протокола, то его использование позволит различным разработчикам обмениваться информацией об используемых ими протоколах, быстро добавлять поддержку новых протоколов в своих продуктах.

Построение универсального декодера различных протоколов на основе их моделей. Для этого на первом этапе потребуется генерация программного протокола на основе данных, содержащихся в модели, и его последующее использование в составе универсальных декодеров, в том числе и мультипротокольных (то есть когда в едином потоке данных перемежаются данные нескольких протоколов). Снижение скорости декодирования примерно на 30%, по сравнению с построением уникального декодера каждого протокола, для большинства практических задач является малозначимым фактором в сравнении с упрощением реализации.

Предоставление в удобном виде для пользователя информации о том или ином протоколе, это может быть дополнением к другим способам описания протокола, например, вербальному или графическому.

Разработка графического «конструктора протоколов». Разработку протоколов с использованием метамодели можно упростить еще больше путем создания конструктора протоколов. Элементы протоколов в нем могут быть представлены графическими элементами, которые пользователь комбинирует в нужном порядке для создания нужного протокола. Генерация XML модели при этом будет выполняться программой-конструктором что позволит исключить ошибки синтаксиса, ускорит и облегчит разработку для начинающих специалистов.

Использование в задачах распознавания неизвестных данных.

Исследование устойчивости протоколов и их декодеров к искажению данных, вычисление времени восстановления, определение требуемых характеристик канала при использовании конкретного протокола и вычисление его числовых характеристик.

Сравнительный анализ однотипных протоколов при моделировании реальных информационных систем и воздействий на них.

Автоматическую упаковку любого протокола в любой транспортный протокол, и решение обратной задачи — устранение транспортного протокола из имеющегося блока данных.

Ведение базы данных протоколов в стандартном формате.

Список прикладных вариантов использования предложенной метамодели может быть расширен и дополнен, приведены только основные направления ее использования.

4. Заключение. В данной работе была предложена метамодель произвольного протокола информационного взаимодействия, позволяющая описывать единым образом протоколы разных типов, со степенью детализации достаточной для их содержательного и сравнительного анализа, построения автоматических декодеров, моделирования и т.д. По сравнению с текстовым описанием (спецификацией) протоколов, наиболее часто используемым сегодня, это является несомненным шагом вперед. Использование метамодели для построения моделей конкретных протоколов позволит выйти на новый уровень абстракции при решении многих прикладных и аналитических задач, например, создать и поддерживать пополняемую базу данных протоколов. В дальнейшем эта модель может быть использована в качестве основы для разработки стандарта описания произвольного протокола. Единственным условием использования метамодели является хорошая структурированность описываемого протокола.

Литература

1. *Назаров А.В., Козырев Г.И., Шитов И.В., Обрученков В.П., Древин А.В., Краскин В.Б., Кудряков С.Г., Петров А.И., Соколов С.М., Якимов В.Л, Лоскутов А.И.* Современная телеметрия в теории и на практике. Учебный курс // СПб.: Наука и техника. 2007. 672 с.
2. *Хантер Д., Кэгл К., Гиббонс Д., Озу Н.а, Пиннок Д., Спенсер П.* Введение в XML // Москва. Издательство ЛОРИ. 2006. 638 с.
3. *Bouzerda T., Marchand-Maille S.* A flexible framework for the development of XML protocols: Applications to MRML // Tech. Rep. no. 03.03. University of Geneva. Switzerland. 2003.
4. *Abdullah I.S., Menasc D.A.* Protocol specification and automatic implementation using XML and CBSE // In: Proc. IASTED Int. Conf. Communications, Internet and Information Technology (CIIT2003). Scottsdale. Arizona. USA. 2003. pp. 191–196.
5. *Basu A., Morrisett G., Von Eicken T.* Promela++: a language for constructing correct and efficient protocols // Proc. INFOCOM 98 17th Annual Joint Conf. IEEE Computer and Communications Societies. San Francisco. USA. 1998. pp. 455–462.
6. *Thees J.* Protocol implementation with Estelle- from prototypes to efficient implementations // Proc. Int'l. Workshop on the Formal Description Technique Estelle. France. 1998.
7. *Leduc G., Germeau F.* Verification of Security Protocols Using LOTOS-method and Application // Computer Communications. 2000. vol. 23 no. 12. pp. 1089-1103.

8. *Bolognesi T., Brinksmaa E.* Introduction to the ISO Specification Language LOTOS // Computer Networks and ISDN Systems. 1987. vol. 14. pp. 25–59.
9. *Huang Y., Ravishankar C.* Cicero: A Protocol Construction Language // Tech. rep. CSE-TR-171-93. Michigan. 1993. pp. 1–39.

References

1. Nazarov A.V., Kozyrev G.I.,Shitov I.V., Obruchenkov V.P., Drevin A.V., Kraskin V.B., Kudryakov S.G.,Petrov A.I., Sokovol S.M., Yakimov V.L., Loskutov A.V. *Sovremennaja telemetrija v teorii i na praktike. Uchebnyj kurs* [Modern telemetry in theory and in practice. training course]. Saint-Petersburg.: Science and Technology. 2007. 672 p. (In Russ.).
2. Hunter D., Cagle K., Gibbons D., Ozu N., Pinnock J., Spencer P. *Vvedenie v XML* [Beginning XML]. Moskva. Izdatel'stvo LORI. 2006. 638 p. (In Russ.).
3. Bouzerda T., Marchand-Maille S. A flexible framework for the development of XML protocols: Applications to MRML. Tech. Rep. no. 03.03. University of Geneva. Switzerland. 2003.
4. Abdullah I.S., Menasc D.A. Protocol specification and automatic implementation using XML and CBSE. In: Proc. IASTED Int. Conf. Communications, Internet and Information Technology (CIIT2003). Scottsdale. Arizona. USA. 2003. pp. 191–196.
5. Basu A., Morrisett G., Von Eicken T. Promela++: a language for constructing correct and efficient protocols. Proc. INFOCOM 98 17th Annual Joint Conf. IEEE Computer and Communications Societies. San Francisco. USA. 1998. pp. 455–462.
6. Thees J. Protocol implementation with Estelle- from prototypes to efficient implementations, Proc. Int'l. Workshop on the Formal Description Technique Estelle. France. 1998.
7. Leduc G., Germeau F. Verification of Security Protocols Using LOTOS-method and Application. Computer Communications. 2000. vol. 23 no. 12. pp. 1089-1103.
8. Bolognesi T., Brinksmaa E. Introduction to the ISO Specification Language LOTOS. Computer Networks and ISDN Systems. 1987. vol. 14. pp. 25–59.
9. Huang Y., Ravishankar C. Cicero: A Protocol Construction Language. Tech. rep. CSE-TR-171-93. Michigan. 1993. pp. 1–39.

Романченко Александр Михайлович — старший преподаватель кафедры систем сбора и обработки информации Военно-космическая академия имени А.Ф. Можайского. Область научных интересов: криптография, информационная безопасность, сетевые технологии. Число научных публикаций — 15. rcrst@newmail.ru; ул. Ждановская, д.13, Санкт-Петербург, 197198, РФ; р.т.: +7(812)237-19-60.

Romanchenko Alexander Mikhailovitch — senior lecturer of the information acquisition and data processing department, Mozhaisky Military Space Academy. Research interests: cryptography, information security, network technology. The number of publications — 15. rcrst@newmail.ru; Zdanovskaya str.13, Saint-Petersburg, Russia, 197198; office phone: +7(812)237-19-60.

РЕФЕРАТ

Романченко А.М. **Обобщенная структурная модель протокола информационного взаимодействия.**

Данная работа посвящена разработке структурной метамодели произвольного протокола информационного взаимодействия, предназначенной для унифицированного представления протоколов разных типов. Она может быть использована в качестве стандарта описания произвольного структурированного протокола. Ее использование позволит разрабатывать, использовать, анализировать различные протоколы и протокольные данные единым способом, что существенно упростит решение многих прикладных и аналитических задач.

При анализе предметной области не были обнаружены стандарты, пригодные для описания произвольного протокола и автоматического использования при решении практических задач. Поэтому предложенное направление можно считать новым и перспективным. Разработанная метамодель не подходит для описания слабо-формализованных и слабо-структурированных протоколов, декодирование которых является контекстно-зависимым. Наибольший эффект можно ожидать от использования метамодели при описании новых прикладных протоколов, и описании протоколов транспортного уровня и ниже.

SUMMARY

Romanchenko A.M. **Generalized Structural Metamodel of Information Interaction Protocol.**

This work is devoted to the development of structural metamodel of any information interaction protocol designed for a unified view of different types of protocols. It can be used as a standard for describing arbitrary structured protocol. Its use will allow to develop, use, analyze different protocols and protocol data in a uniform way, which will significantly simplify the solution of many applications and analytical tasks. In the analysis of the subject area standards have not been found suitable for describing arbitrary protocol and automatic use in solving practical problems. Therefore, the proposed course can be considered as a new and promising.

Designed metamodel is not suitable to describe weakly formalized and weakly structured protocol, decoding of which is context-sensitive. The greatest effect can be expected from the use of meta-model in the description of the new application protocols, and the description of transport layer protocols and lower.