электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 681.5

Этапы разработки ПО и решение задач сбора технических данных конечного Интернет – пользователя

***Гонсалес Х.К.**, студент магистратуры*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Галкин В.А., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
galkin@bmstu.ru

1. Введение.

Автоматизированная система обработки технических данных конечного Интернет – пользователя, позволит произвести контроль качества услуги, оговоренной соглашением между оператором и пользователем, используя статистический анализ специфических показателей. Постановка задачи о разработки данной системы была реализована в статье [1], где была определена основная структура системы, описаны этапы, построены функциональная и имитационная модели системы, и представлены результаты моделирования.

В данной статье представляется задача определения конкретной структуры системы, а именно, следующей ее составной части: разработка программного обеспечения для сбора данных конечного пользователя. Для этого, необходимо реализовать процесс планирования и проектирования программного обеспечения. В анализе используется классический жизненный цикл процесса разработки ПО. Выделяются главные теоретические основы, реализуется декомпозиция системы на подсистемы и описывается структура данных, таким образом, определяются все необходимые потребности для следующего этапа проекта.

2. Основные определения в разработке ПО.

Классический жизненный цикл.

В процессе разработки надежной и эффективной системы, должны быть определены три основных фактора:

1. Методы разработки: анализ системных и программных требований, планирование и оценка проекта, проектирование алгоритмов, структур данных и программных структур, кодирование, тестирование, сопровождение.
2. Средства, обеспечивающие автоматизированную поддержку методов.
3. Процедуры, соединяющие методы и утилиты в непрерывной цепочке разработки.

Таким образом, процесс реализации системы, будет состоять из последовательности шагов, использующих методы, утилиты и процедуры. В данной статье применяется пример классического подхода - модель, в которой разработка рассматривается как последовательность этапов. Переход на следующий этап происходит после полного завершения работ на текущем этапе.

Этапами разработки в классической модели являются:

- системный анализ: определение требований ко всем системным элементам и назначение подмножества этих требований;
- анализ требований: уточняются и детализируются функции, характеристики и интерфейс программного обеспечения;
- проектирование: создается представление архитектуры, модульной и алгоритмической структуры, структуры данных, входного и выходного интерфейса.
- кодирование: перевод результатов проектирования в текст на языке программирования.
- тестирование: выполнение программы для выявления дефектов в функциях, логике и форме реализации программного продукта.
- сопровождение: внесение изменений в эксплуатируемое ПО в целях исправлении ошибок и адаптация ПО.

Первые два этапа являются составляющими планирования проекта.

Полное определение показано на рисунке 1.

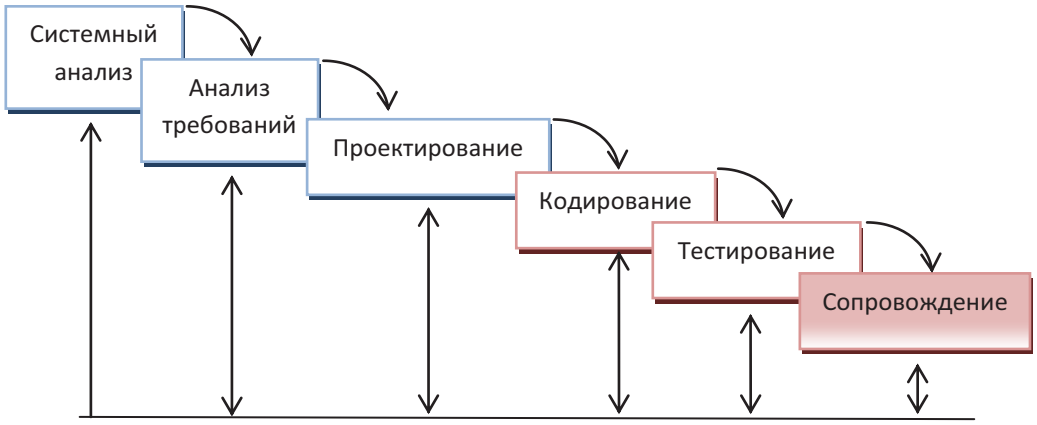

Рис. 1. Классический подход к разработке ПО

3. Планирование проекта. Анализ.

Чтобы реализовать планирование проекта, устанавливаются его цели и проблемная область, альтернативные решения, технические и управленческие ограничения (реализуется системный анализ и анализ технических требований). Общая диаграмма системы представлена на рисунке 2.

Система, по требованию заказчика, содержит четыре этапа:

- сбор данных,
- передача и прием данных,
- обработка данных,
- статистические отчеты и мониторинг.

Рис. 2. Общая схема системы обработки технических данных конечного Интернет –

пользователя

В статье [1], были построены функциональная и имитационная модель системы. Целью настоящего проекта является разработка программного обеспечения для сбора технических данных конечного Интернет – пользователя. В данном анализе, выполняется планирование и проектирование разработки ПО.

4. Выявление технических требований.

После согласования с заказчиком, выделяются технические требования для программного обеспечения, установленного на компьютере пользователя (табл. 1). Последовательность исполнения алгоритма определена столбцом «No.».

На рисунке 3, представлены диаграммы действий и времени, используя Унифицированный Язык Программирования (*UML*), для системы на этапе сбора данных. Программа работает в непрерывном режиме, собирая и отправляя данные, однако допускается остановка исполнения при необходимости.

Таблица 1

Подробные определения технических требований к ПО на этапе сбора и передачи данных (со стороны ПК пользователя)

Этапы	No.	Описание требования	Технические данные (содержание кода)	Дополнительная информация
Сбор данных	1	Технические данные ПК	Приватный и публичный IP адреса, *MAC* адрес, операционная система, подключенное устройство	Для идентификации, дифференцирования и отслеживания статуса каждого клиента системы
	I	Пропускная способность	Сбор скорости передачи в бит/сек в зависимости от направления передачи (вверх/вниз). Выполняется непрерывно в режиме реального времени.	Гистограмма распределения скорости во времени (в зависимости от максимальной скорости)
	2	Отсутствие / обрыв доступа	Сбор данных, когда доступ к Интернет-услуге невозможен (время начала и конца недоступности, и сетевая ошибка)	
	3	Общие ошибки сети	Сбор общих ошибок, влияющих на качество услуг, представляемых	

			пользователю	
	4	Задержка	Измерение задержки к определенным серверам (мсек), с помощью тестов на *ICMP* протоколе	Хосты для проверки должны быть установленными центральным администратором (на сервере)
	-	Пользовательский интерфейс	Интерфейс	Журнал / статистика передачи, информация о системе, временное отключение сбора / передачи
Передача данных	5	Шифрование и запись данных	Проверка целостности данных и запись в отдельных файлах	Файлы должны быть временно записаны на устройстве клиента
	6	Автоматическая передача	Односторонняя передача генерированного файла (через Интернет), каждые 30 мин	После передачи, файлы удаляются с ПК пользователя

После анализа технических требований, так же выводятся следующие дополнительные условия:

- начальное действие происходит после установки программы на ПК пользователя, или при ее запуске (по умолчанию, после запуска ПК). Должен быть создан установочный пакет ПО.
- сбор данных о пропускной способности Интернет – услуги должен вестись непрерывно, чтобы обеспечить эффективность системы, так как скорость передачи зависит от окна времени использованного для расчета.

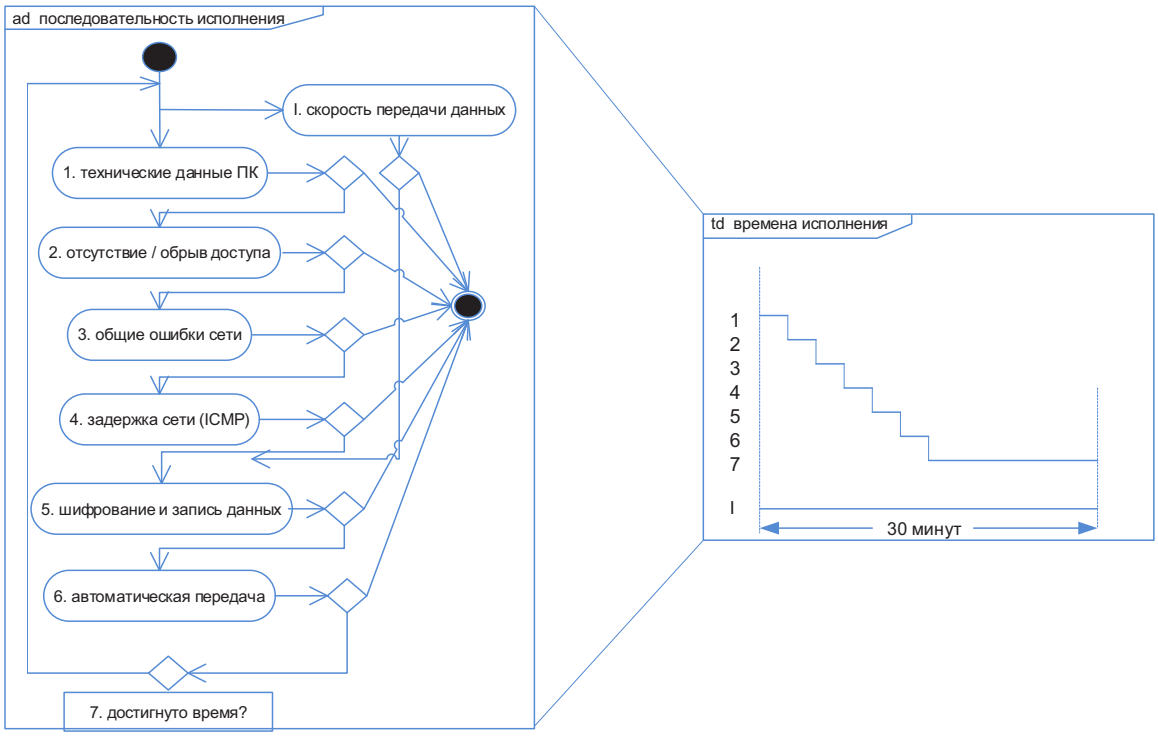

Рис. 3. Диаграммы действий и диаграмма времени (*UML*) для программного обеспечения на ПК пользователя

- остальные параметры не критичны с точки зрения статистики и могут быть собраны один раз перед передачей на сервер, хотя допускается сокращение окна при сборе данных отсутствия/обрыва доступа (3) и общих ошибок сети (4). Зависит от технического решения принятого для сбора данных.
- технические данные собираются в шифрованный файл. Файлы удаляются после передачи на сервер. Структура файла должна быть уникальной. Таким образом, система на этапе обработки данных не зависит от ОС установленной на ПК пользователя (является кроссплатформенной). Тем не менее, на этапе сбора данных необходимо разработать ПО, зависимое от установленной оперативной системы.
- нужно произвести анализ последствия установки модуля сбора данных с точки зрения ресурсопотребления. Модуль не должен оказывать заметного влияния на работу приложений пользователя.
- дополнительные решения (не входящие в первую версию ПО):

- автоматическое определение обновлений программного обеспечения, сообщение пользователю (через интерфейс), и последовательное установление при соглашении пользователя,
- аутентификация клиента в системе, при установке ПО и передачи данных на сервер.

5. Решение технических задач ПО на этапе сбора данных.

5.1. Выбор ОС для разработки.

На этапе сбора данных разрабатываемое ПО зависит от установленной у конечного пользователя операционной системы.

Анализ ситуации на рынке ОС для ПК показал, что согласно источникам [3], [4], [5], [6], на январь 2014 года, большинство персональных и корпоративных устройств работают на ОС *Windows* (65 % - 81 %, в зависимости от анализа), *Apple ОС – OS X / iOS* (10 % - 20 %), и *Unix* подобных ОС – *Linux* / *Android* (10 % - 14 %).

В связи с этим, разработка начального программного обеспечения ограничится на ОС *Windows*.

Так же, ПО должно гарантировать свое исполнение на оперативных системах *Windows XP*, *Windows* 7 и *Windows* 8 (действующих на более чем 95% устройств с *Windows*, соответственно с вышеупомянутыми исследованиями).

5.2. Выбор технологии для извлечения технических данных.

После выбора оперативной системы для разработки ПО, нужно определить технологию, работающую под ОС, которая бы обеспечивала извлечение данных в режиме реального времени.

Windows Management Instrumentation (*WMI*) - инструментарий управления Windows, одна из базовых технологий для централизованного управления и слежения за работой различных частей компьютерной инфраструктуры под управлением платформы *Windows*. Данная технология, адаптированная под стандартом *WBEM*[1], на основе которого лежит идея создания универсального интерфейса мониторинга и управления различными системами и компонентами распределенной информационной

[1] *Web-Based Enterprise Management*, стандартная технология для доступа к информации управления в корпоративной среде.

среды предприятия с использованием объектно-ориентированных идеологий и протоколов *HTML* и *XML*. На рисунке 4 показана архитектура технологии, компоненты ее состава.

В основе структуры данных в *WBEM* лежит *Common Information Model* (*CIM*) - общая информационная модель, реализующая объектно-ориентированный подход к представлению компонентов системы. *CIM* является расширяемой моделью, что позволяет программам, системам и драйверам добавлять в неё свои классы, объекты, методы и свойства. *WMI*, основанный на *CIM*, также является открытой унифицированной системой интерфейсов доступа к любым параметрам операционной системы, устройствам и приложениям, которые функционируют в ней. Важной особенностью *WMI* является то, что хранящиеся в нём объекты соответствуют динамическим ресурсам, то есть параметры этих ресурсов постоянно меняются, поэтому параметры таких объектов не хранятся постоянно, а создаются по запросу потребителя данных.

Общая безопасность в *WMI* реализуется на уровне операционной системы, а дополнительная политика безопасности основана на уровнях пространств имен и протокола *DCOM*. Если пользователь не имеет права выполнять действие через операционную систему, он не сможет это сделать и через *WMI*.

Извлечением данных низкого уровня из управляемого объекта[2] занимается провайдер *WMI*. Провайдер, это *COM* объект[3], состоящий из файла *DLL* и файла управляемый объектным форматом (*MOF*), предоставляя данные на самом низком уровне и определяя классы *WMI*.

Провайдеры передают эти данные на компонент управляющий объектами CIM для их интеграции и интерпретации. Этот компонент оперативной системы состоит из сервиса *WMI* (*winmgmt*) и так называемый «репозиторий» (хранилище свойств объектов *WMI*). Так как *WMI* построен по объектно-ориентированному принципу, то все данные представлены в виде объектов и их свойств и методов. Все классы группируются в пространства имен (например, *root\default, root\cimv2*), которые

[2] Физический или логический управляемый ресурс (жесткий диск, сетевой адаптер, система управления базами данных, операционная система, процесс или сервис), который представлен в виде экземпляра класса WMI.

[3] *Component Object Model* - технологический стандарт *Microsoft*, предназначенный для создания программного обеспечения на основе взаимодействующих компонентов, каждый из которых может использоваться во многих программах одновременно. Стандарт воплощает в себе идеи полиморфизма и инкапсуляции объектно-ориентированного программирования.

иерархически упорядочены и логически связаны друг с другом по определенной технологии или области управления.

Извлечение данных из *WMI* выполняется через приложение или сценарий[4] (управленческие приложения). Сервис *WMI* выступает в качестве посредника между провайдерами, управленческими приложениями и репозиторием.

Для обращения к объектам *WMI* используется специфический язык запросов *WMI Query Language* (*WQL*), который является одним из разновидностей *SQL*. Основное его отличие от *ANSI SQL* — это невозможность изменения данных, то есть с помощью *WQL* возможна лишь выборка данных с помощью команды *SELECT*. Управленческое приложение может запросить, перечислить данные, запустить методы провайдера или подписаться на события.

Классы *Win32* технологии *WMI*, контролируют и управляют системными механизмами и их особенностями, такие как процессы, установленные приложения (*software*), данные операционной системы, производительность.

[4] От английского *script.*

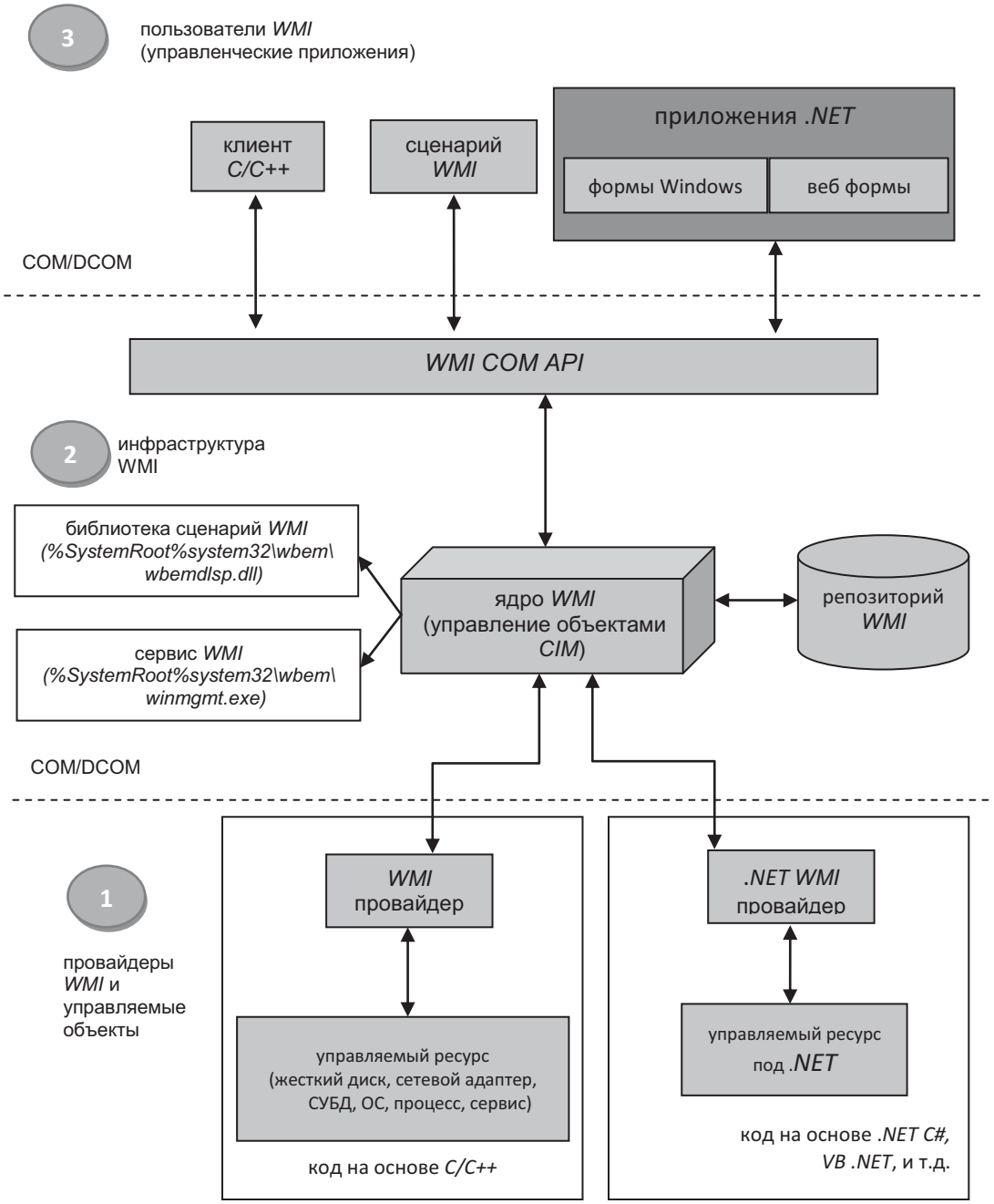

Рис. 4. Архитектура *WMI* (*Windows Management Instrumentation*)

5.3. Сбор технических данных ПК.

Данный подмодуль ПО, должен собирать информацию о дате и времени, подключенном устройстве (тип, модель, марка, протокол подключения), *MAC* адрес подключенного устройства, с целью идентификации клиента, локальный и публичные *IP* адреса. Публичный *IP* адрес, будет полученный от внешнего сайта указанного

сервером. Указатели внешних сайтов для проведения определенных функций будут встроены в файл задач (ФАЗ).[5]

Для извлечения технических данных подключенного в Интернет устройства, используется класс *Win32_NetworkAdapter* (свойства *AdapterType*, *MACAddress*, *SystemName*).

Для извлечения данных версии ОС, используется класс *Win32_OperatingSystem* (свойства *BuildNumber*, *ProductType*, *OSLanguage*, *ServicePackMajorVersion*, *ServicePackMinorVersion*, *Caption*).

Остальные данные не нуждаются в особом способе для сбора и будут зависеть от избранного языка и среды разработки ПО.

5.4. Сбор пропускной способности.

Чтобы получить данные о пропускной способности в режиме реального времени, необходимо учитывать проблему прерывистого информационного трафика обычного пользователя.

Средние Интернет тестеры пропускной способности вычисляют скорость после передачи одного файла на конкретный сервер. Пропускная способность выводится из простого деления объема файла (в байтах) на использованное для его передачи время (в секундах). При этом тестеры предупреждают, о необходимости убедится в отсутствии активных закачек в ПК пользователя перед началом теста. Эти ограничения переносят ответственность за эффективность измерения на сторону конечного пользователя, а так же адресуют достижение этого технического показателя на определенные случайные измерения. Таким образом, рассматривается решение, которое представляет собой более эффективный результат (в сокращенном времени и постоянно работающий).

WMI представляет возможность доступа к счетчикам производительности системы (*Performance Counters*). Эти счетчики, используются для предоставления информации поведении в режиме реального времени операционной системы или приложения, службы или драйвера. Доступ к данным собранные счетчиками (через

[5] Первоначально рассматривается уникальный ФАЗ, передан при установлении ПО. В список дополнительных решений не входящие в первую версию ПО, включается автоматическое определение и передача обновленного ФАЗа.

ADVAPI32.DLL) происходит напрямую, классами WMI «счетчиков производительности» (*Performance Counters Classes*).

Таким классом является *Win32_PerfRawData_ Tcpip_NetworkInterface*. Для вычисления пропускной способности выбираются свойства *BytesReceivedPerSec*, *BytesSentPerSec*, *TimeStamp_PerfTime*, *Frequency_PerfTime*. Следует прояснить значение этих свойств:

- *BytesReceivedPerSec*, *BytesSentPerSec*: количество принятых/отправленных байтов после запуска ПК определенного устройства, и являются кумулятивными.
- *TimeStamp_PerfTime*: мгновенное определение времени, в частотных единицах (*ticks*).
- *Frequency_PerfTime*: количество частотных единиц (*ticks*), которые происходят за одну секунду. Константа, не изменяется во времени.

Таким образом, чтобы вычислить пропускную способность загрузки данных через подключенное в Интернет устройство, стоит извлечь свойства в двух мгновениях $t_1 - t_0$ (где Δt, например, одна секунда), и использовать формулу:

$$Thr\_Rx = \frac{\mathrm{BytesReceivedPerSec}_1 - \mathrm{BytesReceivedPerSec}_0}{(\mathrm{TimeStamp\_PerfTime}_1 - \mathrm{TimeStamp\_PerfTime}_0) \Big/ \mathrm{Frequency\_PerfTime}} [\text{байт/сек}]$$

Аналогично вычисляется пропускная способность передачи информации в Интернет (*Thr_Tx*). Подробное описание процесса во времени объясняется на рисунке 5.

Сбор технических данных для последующего вычисления происходит непрерывно. Расчет выполняется в каждом цикле сбора (1 сек), данные записываются в определенном массиве. Так как помимо пропускной способности в обоих направлениях (вверх/вниз), требуется вычисление гистограммы распределения скорости во времени, выбирается пятиминутное окно, в котором определяются все данные, записываются, и производится сброс использованных массивов. Это делается, чтобы обеспечить минимальную загрузку ПК пользователя (является требованием системы).

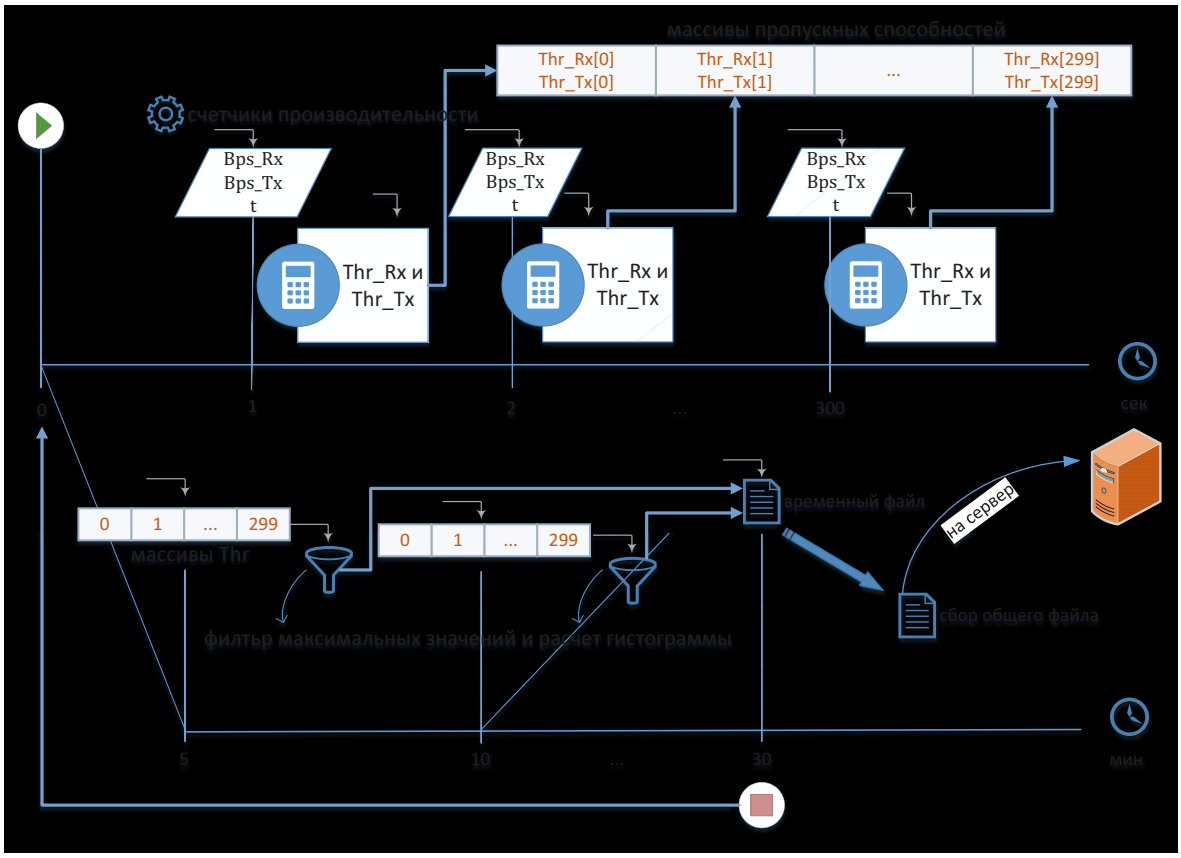

Рис. 5. Процесс сбора данных для вычисления пропускной способности конечного Интернет – пользователя

5.5. Сбор отсутствия / обрыва доступа и общих ошибок сети.

В настоящем подмодуле, извлекаются общие сообщения состояния Интернет – соединения. В случае обнаружения поведения отклоняющегося от нормы, данные сохраняются для передачи на сервер. В данном случае выбранные классы, которые получают и определяют технические показатели состояния Интернет – связи (табл. 2).

Алгоритм ПО, выполняя периодическую проверку данных, записывает присутствующие сообщения о нарушении порядка доступа к услуге и общие полученные ошибки.

Таблица 2

Классы *WMI* предназначены для анализа обрыва доступа и общих ошибок сети на этапе сбора данных

класс WMI	**свойства класса**
Win32_NetworkAdapter	- *LastErrorCode* - *NetConnectionStatus*
Win32_PerfRawData_Tcpip_NetworkInterface	- *PacketsOutboundErrors* - *PacketsReceivedErrors*
Win32_NTLogEvent	- *EventCode* - *Message* - *TimeGenerated*
Win32_PerfRawData_PerfNet_Browser	- *IllegalDatagramsPerSec* - *ServerAnnounceAllocationsFailedPerSec* - *MissedServerListRequests* - *MailslotReceivesFailed*

5.6. Задержка (*ICMP*).

Чтобы получить задержку сети, используется протокол межсетевых управляющих сообщений (*ICMP*). Адреса серверов для проведения тестов будут указаны через ФАЗ. Последующее вычисление будет исполняться классом *Win32_PingStatus* технологии *WMI*.

Выбранные свойства: *Address* (адрес тестового сервера), *StatusCode* (ответ на отправленный *ping*), *BufferSize*(объем отправленного пакета), *ResponseTime* (время *ICMP* ответа), *TimeToLive* (промежуток времени для окончания теста, уменьшается на 1 при каждом «прыжке» через очередное сетевое устройство»), и *ResponseTimeToLive* (тот же *TimeToLive*, но в обратном пути). Данные свойства выдают напрямую требованные технические данные, без дополнительных вычислений.

5.7. Сбор данных.

После сбора данных в каждом подпроцессе (1-4) этапы, происходит соединение, шифрование данных, и последовательная передача на сервер. На рисунке 6 показана общая схема описанного процесса для выполнения ПО установленного в компьютере конечного пользователя.

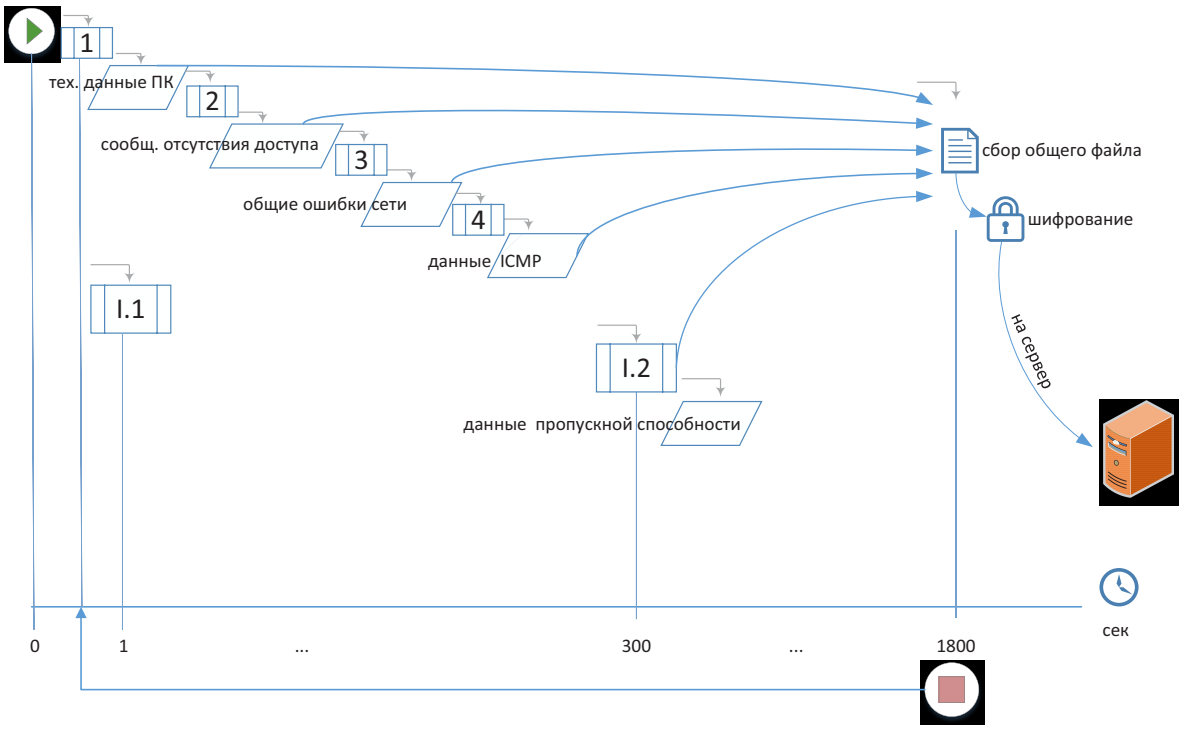

Рис. 6. Завершенный процесс сбора данных перед передачей на сервер

В общем файле, данные вводятся по порядку 1 – 2 – 3 – 4 – I (строго соблюдая последовательность данных). Между данными, включается знак разделитель «|», для определения на сервере на этапе обработки перехода к следующему полю общей структуры данных.

6. Заключение.

После определения основной структуры системы обработки технических данных конечного Интернет – пользователя, предстоит задача проектирования системы, структурное планирование разработки программного обеспечения, установленного на ПК пользователя, на этапе сбора технических данных.

В данной статье, ведется краткое описание этап разработки программного обеспечения, а так же реализуется планирование (системный анализ и анализ требований заказчика) и общее проектирование (представление модульной архитектуры и алгоритмической структуры) программного обеспечения.

Далее, решаются технические задачи для правильного сбора данных (оперативная система первоначальной разработки, технология извлечения данных). Выбирается и реализуется анализ технологии *Windows Management Instrumentation*

(*WMI*), описываются классы и формулы применения для вычисления требованных данных.

В конце статьи, строится общая структура процесса сбора данных, выводятся временные диаграммы процесса, и описывается структура данных общего файла для передачи на сервер.

Список литературы.

1. Гонсалес Х.К., Галкин В. А. Постановка задачи о разработке автоматизированной системы обработки технических данных конечного интернет-пользователя республики Эквадор. Инженерный вестник (МГТУ им. Н.Э.Баумана). Электронный журнал. 2013 .- № 10. http://engbul.bmstu.ru/doc/640684.html.
2. Орлов С.А. Технологии разработки программного обеспечения: Учебник. СПб.: Питер, 2002. — 464 с.
3. *W3Schools*. Статистики платформ оперативных систем. 2014. Режим доступа: http://www.w3schools.com/browsers/ browsers_os.asp (дата обращения 24.02.2014).
4. *NETMARKETSHARE*. Доля рынка операционных систем персональных компьютеров. 2014. Режим доступа: http://marketshare.hitslink.com/operating-system-market-share.aspx? qprid=10&qpcustomd=0 (дата обращения 24.02.2014).
5. *StatCounterGlobalStats*. Первые 8 оперативных систем в январе 2014. 2014. Режим доступа: http://gs.statcounter.com/#all-os-ww-monthly-201401-201401-bar. (дата обращения 24.02.2014).
6. Бочкарёв В. Администрирование с помощью *WMI*. 2012. Режим доступа: http://www.sysengineering.ru/Administration/Administration UsingWMI.aspx. (дата обращения 25.02.2014).
7. Корпорация Майкрософт. Архитектура *WMI*. 2014. Режим доступа: http://msdn.microsoft.com/en-us/library/aa394553(v=vs.85) .aspx. (дата обращения 25.02.2014).
8. Корпорация Майкрософт. Классы *Win32*. 2014. Режим доступа: http://msdn.microsoft.com/en-us/library/aa394084(v=vs.85).aspx. (дата обращения 26.02.2014).

9. Корпорация Майкрософт. Сценарии *WMI*: Часть 1. 2014. Режим доступа: http://msdn.microsoft.com/en-us/library/ms974579.aspx. (дата обращения 26.02.2014).
10. Корпорация Майкрософт. О счетчиках производительности. 2014. Режим доступа: http://msdn.microsoft.com/en-us/library/aa371643 (v=vs.85).aspx. (дата обращения 26.02.2014).