электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Анализ и синтез программного обеспечения сбора технических данных конечного Интернет – пользователя

 # 11, ноябрь 2014

Гонсалес Х. К., Галкин В. А.

УДК: 681.5

Россия, 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 [6], где была определена основная структура системы, описаны этапы, построены функциональная и имитационная модели системы, и представлены результаты моделирования.

В статье [7] дано описание этапа разработки программного обеспечения (ПО) на этапе сбора технических данных с изначальным анализом требований заказчика, представлением модульной архитектуры и алгоритмической структуры ПО, а также решением технических задач сбора данных (оперативная система, технология извлечения данных).

В данной статье представлены подробный анализ и синтез компонентов ПО, устанавливаемого на ПК конечного пользователя, на этапе сбора технических данных, анализ начальной оптимизации, результаты реализации и тестирования данного ПО.

2. Описание компонентов ПО.

После подробного описания требований автоматизированной системы обработки технических данных конечного Интернет – пользователя [6], выявляются четыре этапа данной системы:

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

Задачей программного обеспечения на этапе сбора данных, а именно, ПО установленного на ПК Интернет – пользователя, является сбор всех технических показателей, ко-

торые по требованию клиента позволяют статистически произвести контроль качества услуги. Эти технические данные были подробно описаны в [7], из них можно выделить:

- **технические данные ПК**: приватный и публичный IP адреса, MAC адрес, операционная система, подключенное устройство,
- **пропускная способность**: Сбор скорости передачи в бит/сек в зависимости от направления передачи (вверх/вниз). Гистограмма распределения скорости во времени.
- **отсутствие/обрыв доступа**: события ОС связанные с Интернет соединением,
- **общие ошибки сети:** общие ошибки передачи данных, а так же, результаты тестов,
- **задержка:** измерение задержки к определенным серверам (мсек), с помощью тестов на ICMP протоколе,

Дополнительными потребностями к программному обеспечению являются:

- **пользовательский интерфейс**: журнал/статистика передачи, информация о системе, временное отключение сбора/передачи,
- **шифрование и запись данных,**
- **автоматическая передача:** односторонняя передача через Интернет сгенерированного файла на сервер (каждые 30 мин).

На этапе анализа разработки ПО, установленного на ПК пользователя, (сбор данных) используют диаграммы действий и времени приведенные в [7]. После этого, конкретно анализируют действия, выполняемые на этапе сбора данных в контексте всей системы. Это способствует созданию надежной и устойчивой архитектуры и облегчает глубокое понимание требований к системе.

Для описания результатов анализа требований к системе разработана модель, которая сохраняет актуальность в течение всего жизненного цикла программного обеспечения.

На рисунке 1, детализирована диаграмма классов анализа системы, используя три стандартных стереотипа классов Унифицированного языка моделирования - UML («сущность», «граница», и «управление»). Пунктирной линией выделены основные задачи, исполняемые ПО на этапе сбора технических данных. Стереотипы «сущность» указывают на технические данные, которые должны быть собраны программой на данном этапе.

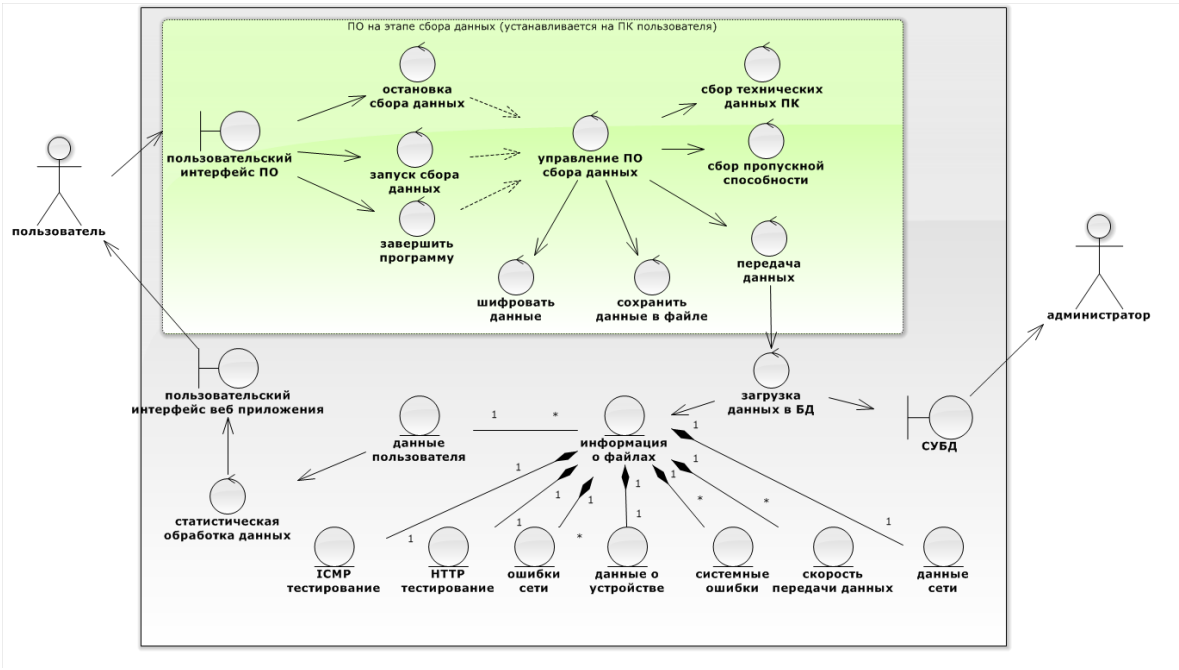

Рис.1 Диаграмма классов анализа системы (UML).

2.1 Детальное определение компонентов реализации системы.

После детального понимания требований к программному обеспечению, следует создать соответствующие исходные данные и базовые точки для последующей реализации. Для этого, при наличии модели анализа, выполним декомпозицию системы на множество управляемых абстрактных частей (классов) и определим порядок их выполнения.

На рисунке 2 представлена диаграмма классов ПО на этапе сбора данных. Диаграмма построена с учетом технических решений, принятых для реализации на основе оперативной системы *Microsoft Windows*, языка программирования C#, технологии для извлечения данных *Windows Management Instrumentation* (WMI) и подробно описанных в [7].

Кратко опишем спроектированные классы и их главные компоненты:

i. **Main_Window:** главный класс в ПО типа Windows Forms C#. Поочередно вызывает отдельные классы в соответствии с процессом сбора данных. Содержит следующие компоненты:
   - **Переменные:** _allResults – содержит все технические данные являющиеся результатом этапа сбора, _throughput - содержит данные скорости передачи пакетов Интернет пакетов.
   - **Методы:** Start() – инициирование программы сбора данных, StopCollection() - временное отключение сбора, RestartCollection() - повторный запуск сбора данных, Minimize() - минимизировать окно пользовательского интерфейса, Close() – остановить выполнение программы и закрыть интерфейс.

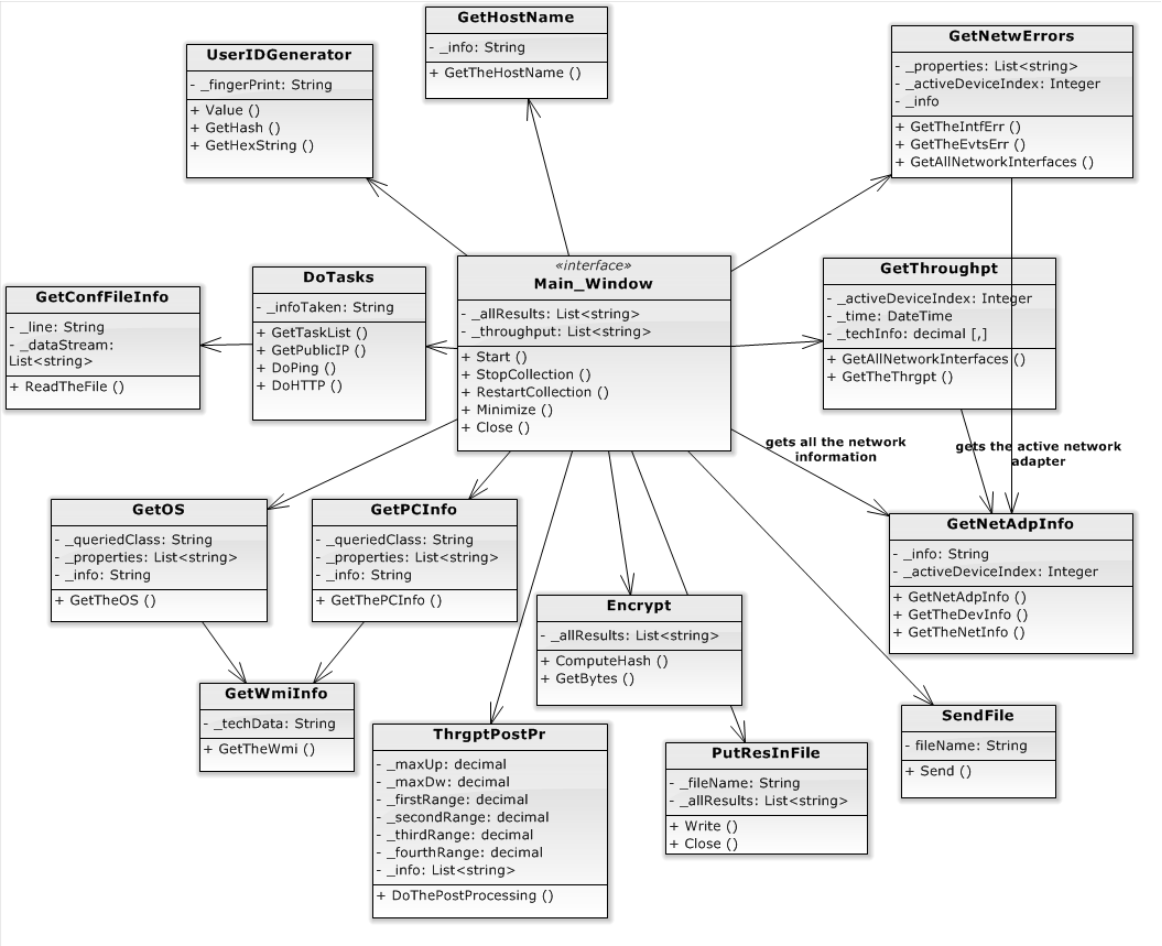

Рис2 Диаграмма классов проектирования ПО (UML).

ii. **UserIDGenerator:** формирует уникальный идентификационный код ПК пользователя. Содержит:

- **Переменные:** _fingerPrint – значение уникального идентификационного кода ПК пользователя.
- **Методы:** Value() – вызывает функции сбора уникальных значений ПК, GetHash() – шифрует данные используя криптографические алгоритмы, GetHexString() – собирает данные кода в определенном формате.

iii. **GetHostName:** получает название хоста пользователя. Содержит:

- **Переменные:** _info – хранит передаваемое название хоста.
- **Методы:** GetTheHostName() – реализует извлечение название хоста.

iv. **GetNetwErrors:** получает общие ошибки сети и события ОС связанные с Интернет соединением. Содержит:

- **Переменные:** _properties – содержит указатели на события собранные на ПК, _activeDeviceIndex – указатель активного сетевого устройства (подключенного в Интернет), _info – результат возвращенный классом (ошибки).

- **Методы:** GetTheIntfErr() – сбор ошибок на сетевом уровне (пакетов), GetTheEvtsErr() – сбор событий ОС связанные с Интернет соединением, GetAllNetworkInterfaces() – сбор данных на сетевом уровне.

v. **GetThroughpt:** реализует сбор скорости передачи данных в бит/сек в зависимости от направления передачи (вверх/вниз). Содержит:

- **Переменные:** _activeDeviceIndex - указатель активного сетевого устройства (подключенного в Интернет),_time – время сбора данных, _techInfo – хранит передаваемые данные.
- **Методы:** GetAllNetworkInterfaces() – сбор данных на сетевом уровне, GetTheThrgpt() – реализует сбор скорости передачи данных.

vi. **GetNetAdpInfo:** Получает данные активного сетевого адаптера. Содержит:

- **Переменные:** _info - передаваемые технические данные сетевого устройства, _activeDeviceIndex – указатель активного сетевого устройства (подключенного в Интернет).
- **Методы:** GetNetAdpInfo() – выдает индекс активного сетевого устройства, GetTheDevInfo() – выдает информацию устройства (MAC и тип устройства), GetTheNetInfo() – выдает информацию сетевого уровня (IP адресов).

vii. **DoTasks:** выполняет технические определенные задачи. Содержит:

- **Переменные:** _infoTaken – передаваемые данные, результаты технических задач (статус, время исполнения, количество передаваемых данных).
- **Методы:** GetTaskList() – получает список задач на выполнение, GetPublicIP() – определение публичного IP адреса через заданный сервер, DoPing() – реализует тест ICMP протокола, DoHTTP() – реализует тест HTTP протокола.

viii. **GetConfFileInfo:** передает технические задачи извлекая из конфигурационного файла (ФАЗ). Содержит:

- **Переменные:** _line – хранит строки данных ФАЗ, _dataStream – хранит список задач в определенной форме для передачи.
- **Методы:** ReadTheFile() – открывает ФАЗ, читает конфигурационные данные, и закрывает ФАЗ.

ix. **GetOS:** получает и передает информацию об операционной системе. Содержит:

- **Переменные:** _queriedClass – WMI класс, из которого извлекаются данные, _properties – WMI свойства, технические данные, _info –результат ОС ПК для передачи.
- **Методы:** GetTheOS() – получает информацию об операционной системе ПК.

x. **GetPCinfo:** получает и передает сведения о компьютере (производитель и модель). Содержит:

- **Переменные:** _queriedClass – WMI класс, из которого извлекаются данные, _properties – WMI свойства, технические данные, _info –результаты сведений о компьютере.

- **Методы:** GetThePCInfo() – получает технические сведения о компьютере.

xi. **GetWmiInfo:** получает WMI информацию, выполняя запросы на основе свойств определенными вызывающим объектом. Содержит:
- **Переменные:** _techData – хранит WMI информацию, на передачу.
- **Методы:** GetTheWmi() – выполняет запрос WQL (WMI Query Language), извлекает данные для передачи.

xii. **ThrgptPostPr:** обрабатывает и передает информацию о скорости передачи данных в заданном формате. Содержит:
- **Переменные:** _maxUp и _maxDw – хранит временные лучшие показатели скорости (вверх/вниз), _firstRange по _fourthRange – пределы скоростей, определяющие гистограмму распределения скорости во времени, _info – результат для передачи.
- **Методы:** DoThePostProcessing () – выполняет обработку информации скорости передачи данных в заданном формате.

xiii. **Encrypt:** обрабатывает и шифрует данные полученные на этапе сбора. Содержит:
- **Переменные:** _allResults – содержит все технические данные являющиеся результатом этапа сбора.
- **Методы:** GetBytes() – преобразует данные в байтовый формат для шифрования, GetHash() – шифрует данные используя криптографические алгоритмы.

xiv. **PutResInFile:** обрабатывает и сохраняет данные в конечном файле для передачи на сервер. Содержит:
- **Переменные:** _allResults – содержит все технические данные являющиеся результатом этапа сбора, _fileName – имя файла для передачи на сервер.
- **Методы:** Write() – открывает и записывает данные в файле, Close() – закрывает файл.

xv. **SendFile:** передает файл на сервер. Содержит:
- **Переменные:** _fileName – имя файла для передачи.
- **Методы:** Send() – передает файл на сервер.

2.2 Предметная область системы.

Подробное описание компонентов системы, позволило определить функциональные характеристики ПО. На данном этапе разработки уже известны функции, которые будет выполнять программа на ПК пользователя. Следующим шагом выполним детализацию ожидаемых результатов в виде предметной области системы. На рисунке 3 представлена диаграмма классов предметной области системы в терминах UML.

Дадим краткое описание предметной области (сущностей):

i. **Users:** таблица, хранящая учетные записи пользователей системы (идентификационный номер, уникальный идентификатор, и данные для авторизации на сайт системы).

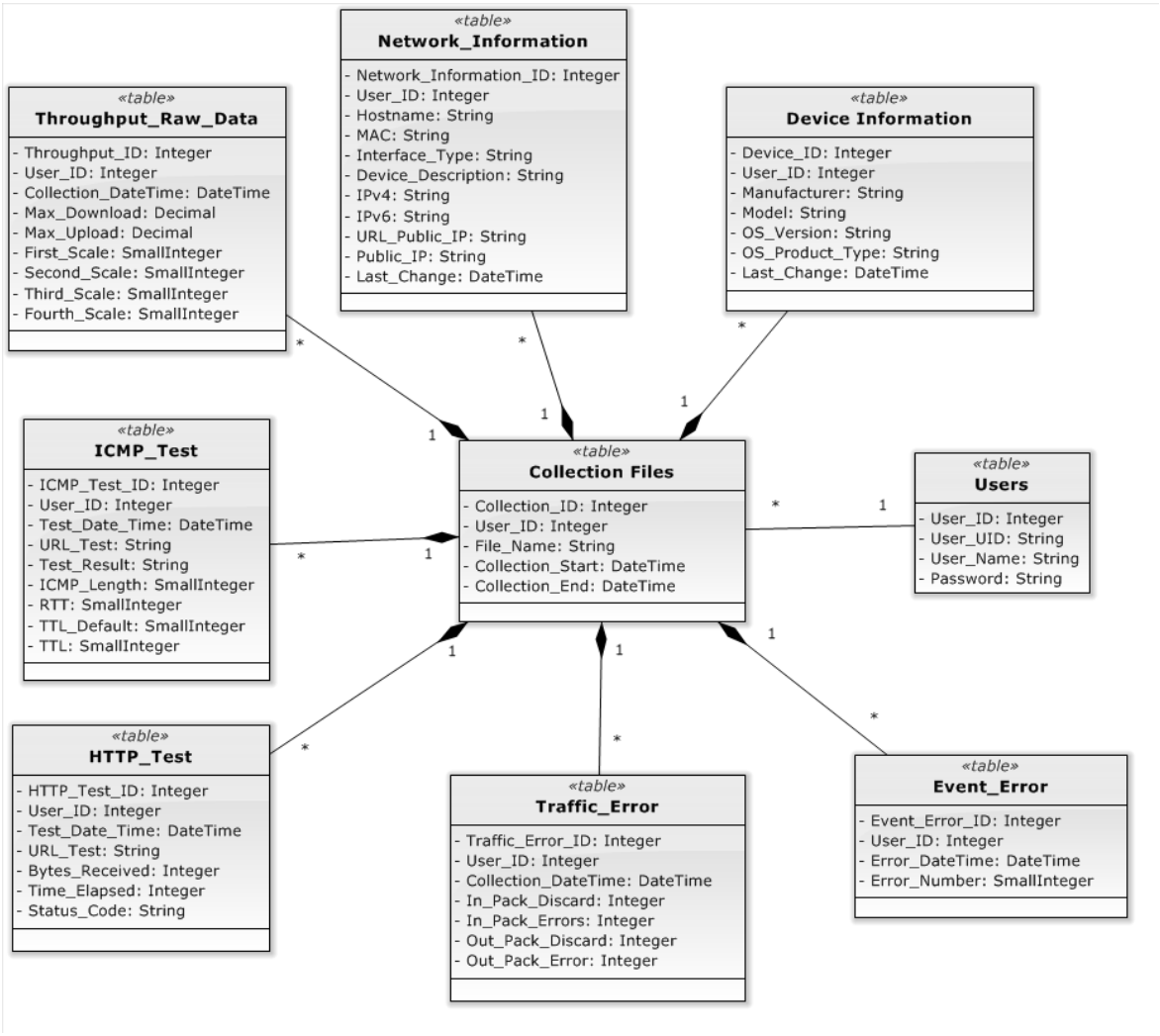

Рис.3 Диаграмма классов предметной области системы (UML).

ii. **Collection_Files:** информация о передаваемых файлах (идентификатор, пользователь, название, и сведения о дате собранной информации).

iii. **Device_Information**: данные устройства подключенного к Интернет – услуге (сведения о компьютере, ОС).

iv. **Network_Information**: данные сетевого уровня (техническая информация сетевого устройства, MAC и IP адреса, публичный адрес).

v. **Throughput_Raw_Data**: данные скорости передачи информации (вверх/вниз) и гистограмма эффективного распределения скорости.

vi. **ICMP_Test**: сведения задачи технического «пинг» - теста, и показатели результата данного теста (количество прыжков, времена выполнения, статус).

vii. **HTTP_Test**: сведения задачи технического HTTP - теста, и показатели результата данного теста (количество переданной информации, времена выполнения, результаты).

viii. **Traffic_Error**: данные ошибок на сетевом уровне (переданных пакетов).

ix. **Event_Error**: данные ошибок на уровне оперативной системы, связанные с Интернет - подключением.

3. Результаты разработки.

После того как было подробно определена логическая организация системы, приступаем к созданию программного продукта, который обеспечивает начальные операционные потребности. Создаем классы и методы, описанные в предыдущем пункте, на языке программирования C#, используя итерационный метод конструирования элементов *Use Case*, включая в каждой итерации:

1. идентификацию реализуемых классов и отношений,
2. определение в классах типов данных сервисных операций,
3. создание текста на языке программирования,
4. тестирование функций реализации продукта.

Результатом разработки определенных классов является программа, выполняющая технические требования установленные клиентом на первоначальном этапе разработке системы. На рисунке 4 представлена диаграмма классов ПО с использованием среды разработки *Visual Studio Professional 2013 (Version 12.0.210005.1 REL).*

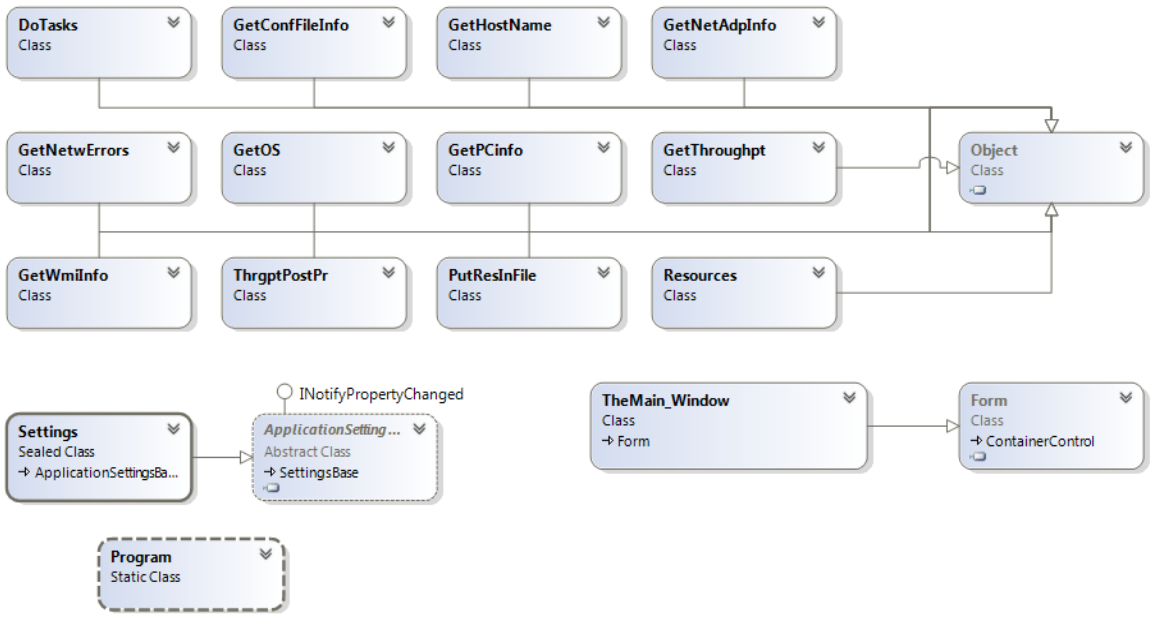

Рис.4 Диаграмма классов ПО (Visual Studio Professional 2013)

3.1 Оптимизация методов на этапе разработки.

По ходу реализации ПО, были идентифицированы и оптимизированы методы, которые по предварительному тестированию оказались менее эффективными. Так, изначальное техническое решение извлечения данных с использованием инструментария управления *Windows* (WMI) было заменено в некоторых классах, в связи с большим временем выполнения, на методы, встроенные в язык программирования C#.

В таблице 1 представлены результаты сравнения времени выполнения WMI и встроенных классов, доказывая принятое решение заменить методологию извлечения данных в некоторых классах.

Анализ результатов показывает, что в большинстве случаев время исполнения WMI на порядок менее эффективно, чем время исполнения классов C#.

Особым случаем является класс GetOS. Хотя время выполнения задачи по методу WMI превышает более чем в 13 раз время выполнения с использованием встроенного класса *System.Environment*, встроенный класс не предоставляет полную необходимую техническую информацию (не выдает значение *product type* для точного определения операционной системы ПК). Поэтому, в данном классе используется метод извлечения технологией WMI.

Таблица 1. Результаты сравнивания выполнения задачи извлечений технических данных, используя WMI и встроенных C# классов.

Класс	**Конкретная задача метода**	**Время выполнения, используя WMI (сек)**	**Встроенный C# класс**	**Время выполнения, используя C# класс (сек)**
GetHostName	получить название хоста	0.512	System.Environment	0.034
GetOS	получить данные о ОС (*version, product type*)	0.420	System.Environment	0.032
GetNetAdpInfo	извлечь данные сетевого уровня (техническая информация сетевого устройства, MAC и IP адреса, публичный адрес)	3 – 7	System.Net. NetworkInformation. NetworkInterface	0.700
DoTasks, метод DoPing()	после отправления пакета ICMP, получить результаты теста (количество прыжков, времена выполнения, статус)	1.426	System.Net. NetworkInformation. Ping, System.Net. NetworkInformation. PingOptions	0.194

Так же, в классе GetPCinfo используется изначально выбранная технология извлечения данных WMI, поскольку полностью предоставляет необходимые технические данные.

3.2 Конечные результаты.

Оперативное тестирование ПО подтвердило наличие всех технических данных требуемых от работы системы. Тестирование было проведено в режиме доступа к Интернет – услуге на ПК реального пользователя. На рисунке 5 показан текстовый файл содержащий результаты сбора технических данных Интернет – услуги на ПК конечного пользователя. Специально удалены конфиденциальные данные, такие как адреса и названия хоста.

Стоит так же описать инициалы, использованные для отображения результатов:

- **DTsc**: время начала сбора данных,
- **Thrp**: данные скорости передачи информации (вверх/вниз) и поминутная гистограмма эффективного распределения скорости (измеряется ежесекундно),
- **HstN:** название хоста,
- **MfMd:** производитель и модель ПК,
- **OpSy:** данные ОС,

- **AdpI:** данные сетевого устройства,
- **IPvX:** приватные IP адреса,
- **InEr:** ошибки сетевого уровня (передача пакетов),
- **PuIP:** публичный IP адрес,
- **ICMP:** показатели и результаты «пинг» теста,
- **HTTP:** показатели и результаты HTTP теста,
- **DTec:** время конца сбора данных.

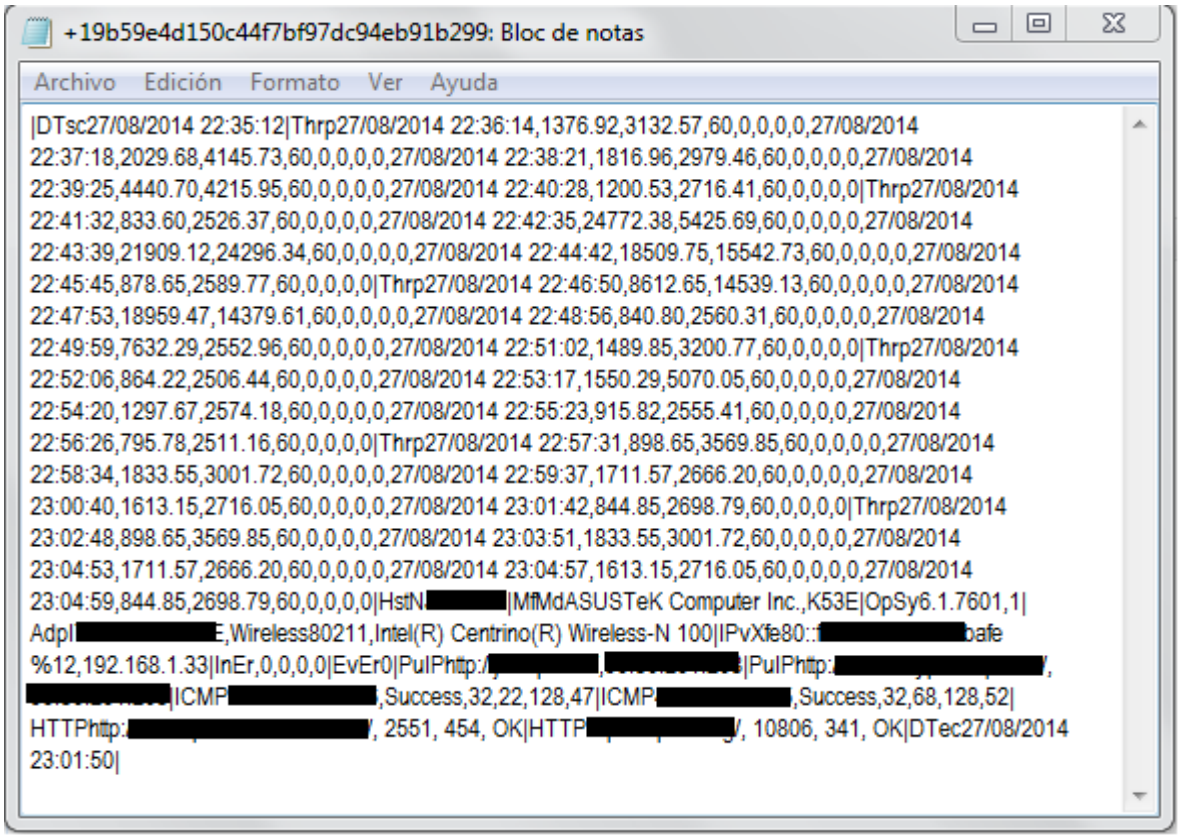

Рис.5 Результат сбора технических данных на ПК пользователя

Заключение

Проведенный анализ позволил детально описать логическую структуру системы и, используя методологический процесс описания функциональных характеристик реализуемого ПО, синтезировать необходимые классы и их главные компоненты для последующей разработки.

Определена предметная область, которая обеспечивает хранение результатов сбора технических данных.

Проведенное тестирование выявило необходимость оптимизации методов на этапе разработки. Показано, как в некоторых случаях, замена изначально выбранной технологии извлечений технических данных от *Microsoft* – WMI на встроенные в язык программирования классы, позволила сократить более чем в 10 раз время выполнения конкретных задач сбора данных.

Реализация ПО на языке программирования C# в среде разработки *Visual Studio Professional 2013* показала состоятельность принятых проектных решений. Реализованное программное обеспечение соответствует требованиям по сбору технических данных в режиме доступа к Интернет – услуге на ПК реального пользователя.

Список литературы

1. Орлов С.А. Технологии разработки программного обеспечения: Учебник. СПб.: Питер, 2002. — 464 с.
2. Якобсон А., Буч Г., Рамбо Д. Унифицированный процесс разработки программного обеспечения. СПб.: Питер, 2002. — 492 с.
3. Арлоу Д., Нейштадт И. UML 2 и Унифицированный процесс: Практический объектно-ориентированный анализ и проектирование. СПб: Символ - Плюс, 2007. – 624 с.
4. Анисимов В. В. Проектирование информационных систем: Учебно-методический материал. 2014. Режим доступа: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema14 (дата обращения 25.09.2014).
5. Корпорация Майкрософт. Классы (C#. Учебник программирования). Режим доступа: http://msdn.microsoft.com/en-us/library/x9afc042.aspx (дата обращения 27.09.2014).
6. 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.
7. Гонсалес Х.К., Галкин В. А. Этапы разработки ПО и решение задач сбора технических данных конечного Интернет – пользователя. Инженерный вестник (МГТУ им. Н.Э.Баумана). Электронный журнал. 2014.- ФС77-51036. http://sntbul.bmstu.ru/doc/699574.html.