ГЛАВА 2

Разработка алгоритма работы автоматизированной системы

На начальном этапе создания автоматизированной системы следует разработать алгоритм ее работы. Здесь стоит отметить, что под алгоритмом работы автоматизированной системы будет подразумеваться алгоритм работы программного обеспечения, выполняющего соответствующие функции. Также стоит уточнить, что в данном контексте в качестве алгоритма будет рассматриваться последовательность инструкций, выполняемых в рамках одного стабилографического измерения – разумеется, стабилографическое исследование может включать в себя более одного измерения.

В соответствии с алгоритмом проведения типичного стабилографического измерения, приведенного в главе 1, а также принимая во внимание выявленные функциональные и нефункциональные требования к проектируемой автоматизируемой системе, приведенные в той же главе, был разработан следующий алгоритм работы системы.

- Предварительные действия: снятие показаний с аппаратно-программного комплекса «Многофункциональное кресло» при помощи персонального компьютера с использованием программного обеспечения StabMed версии 2.09 и выше, результат – стабилограмма (сигнал).

- Запуск проектируемой автоматизированной системы обработки результатов стабилографических измерений.

- Формирование и отображение графического пользовательского интерфейса автоматизированной системы.

- Получение системой данных (сигналов) из StabMed.

- Математическая обработка полученных сигналов.

- Анализ обработанных данных на предмет наличия аномалий в сигнале, которые предположительно могут являться маркерами патологий нервной системы и/или опорно-двигательного аппарат испытуемого.

- Визуализация обработанных стабилограмм и результатов их анализа.

- Классификация данных как носящих либо не носящих патологический характер экспертом (исследователем-физиологом).

- Сохранение классифицированной, обработанной предусмотренными системой математическими методами стабилограммы.

- Останов автоматизированной системы.

Проектирование структуры автоматизированной системы

На основании приведенного выше алгоритма можно произвести декомпозицию будущей автоматизированной системы на автономные подсистемы, выполняющие определенную функцию в рамках алгоритма работы. Таким образом проектирование и реализация значительно упрощаются. Данный подход также упрощает и повышает эффективность тестирования системы, поскольку появляется возможность начать тестирование на более раннем этапе проектирования и реализации. Большей эффективности также будет способствовать то, что на ограниченный функционал подсистемы единовременно потребуется меньше тестов, что облегчит обнаружение дефектов.

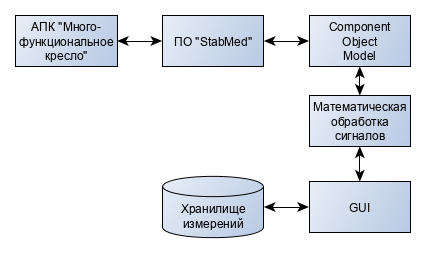


Рис. Структурная схема автоматизированной системы

На основании алгоритма работы, можно выделить следующие составные части автоматизированной системы (структурная схема представлена на рис. ):

- интерфейс Component Object Model;

- модуль математической обработки сигналов;

- графический пользовательский интерфейс;

- хранилище измерений.

Разработка интерфейса по стандарту Component Object Model

Component Object Model (COM, компонентная объектная модель) - это стандарт интерфейса для программных компонентов, созданный компанией Microsoft. Он используется для создания объектов межпроцессного взаимодействия с использованием большого количества различных языков программирования. COM является основой для нескольких других технологий и платформ Microsoft, включая OLE, OLE Automation, Browser Helper Object, ActiveX, COM +, DCOM, DirectX и некоторых других. Суть COM - это независимый от языка способ реализации объектов, который можно использовать в средах, отличных от той, в которой они были созданы, даже за пределами машин. Качественно написанные компоненты COM позволяют повторно использовать объекты при отсутствии сведений об их внутренней реализации, вынуждая разработчиков компонентов предоставлять четко определенные интерфейсы, которые отделены от реализации.

Концепция объектов в COM похожа на концепцию объектно-ориентированного программирования, являясь, по сути, ее более универсальным вариантом. При реализации Component Object Model объекты располагаются в отдельных исполняемых модулях. Такими модулями могут быть либо файлы Dynamic-link library (DLL), либо непосредственно EXE-приложение.

При этом Component Object Model не является ни языком программирования, ни библиотекой, ни компилятором – это набор правил проектирования программ, которые будут вести себя определенным образом и иметь специфические возможности.

Основу модели COM составляют три принципа:

1. Независимость от контекста и уникальность компонентов: компоненты с одинаковым способом обращения имеют одинаковый смысл и назначение.
2. Инкапсуляция: внутреннее устройство COM-объектов скрыто для обеспечения независимости от языков программирования.
3. Доступность и портативность: необходимый компонент можно быстро найти динамически без привязки к файловой системе.

Таким образом преимуществом Component Object Model является наличие у каждого разработчика возможности кроссплатформенного обмена информацией между разрабатываемыми модулями, без привязки к средствам реализации и языкам программирования.

В модели COM взаимодействие между компонентами выстраивается по схеме «Клиент-сервер»: существует исполняемый модуль, в функционал которого включено исполнение некой инструкции, и приложение-инициатор, которому необходимо исполнить данную инструкцию, но которое «не знает», как это сделать, «зная» при этом о существовании исполняемого модуля и обращаясь к нему при возникновении такой необходимости; первый компонент будет являться «сервером», а второй – «клиентом». Component Object Model предусматривает два типа серверов: внутренние, представляющие собой исполняемые программы, и внешние, являющиеся динамически компонуемые библиотеками.

Приложение связывается с компонентом при помощи его имени. Для обеспечения отсутствия коллизий имен на разных машинах и в продуктах разных производителей COM-компоненты именуются при помощи 16-байтных чисел, вычисляемых специальной хэш-функцией. Эти номера носят название GUID – Globally Unique Identifier.

В силу соблюдения принципа инкапсуляции, реализация COM-компонента закрыта для стороннего пользователя (и, соответственно, приложения) – известен лишь поддерживаемый компонентом функционал. Доступ к этому функционалу осуществляется при помощи интерфейса, своеобразного контракта между пользователем и COM-объектом, который по сути представляет собой список указателей на реализованные функции. Подобно компонентам, COM-интерфейсы также имеют свой уникальный идентификатор – IID (Interface Identifier).

Связь между клиентом и сервером обеспечивает операционная система. В системном реестре Windows хранится информация о COM-сервере: его название, GIUD, уникальный номер компонента CLSID (Class Identifier).

Программное обеспечение «StabMed» реализует стандарт Component Object Model, выступая в роли COM-сервера. Посредством поддерживаемого интерфейса ISignalExporter сторонние приложения могут получать информацию о сигналах, импортировать сами сигналы, а также информацию о пациентах, в ходе обследования которых эти сигналы были получены. Полное описание реализации стандарта Component Object Model данным ПО приводится в Приложении 1.

В рамках данной работы при проектировании автоматизированной системы задача состоит в написании программного модуля, который бы обеспечивал программе доступ к нужным данным. Поскольку программная часть системы реализуется на языке Python, необходимо было найти реализацию компонентов Component Object Model на данном языке программирования. С этой целью была выбрана библиотека pywin32. Помимо наличия реализации всех требуемых для создания COM-компонентов функций, библиотека примечательна наличием подробного описания принципов и методов работы как в виде сопроводительной документации, так и в виде книги Python Programming on Win32 [6], материалы из которой и были использованы при реализации данного компонента автоматизированной системы.

В StabMed реализован внутренний COM-сервер, то есть сервер внутри EXE-приложения. Об этом говорит свойство InprocServer32 в описании соответствующей библиотеки типов, хранящемся в системном реестре Windows. Таким образом, в рамках взаимодействия с данным сервером необходимо зарегистрировать библиотеку типов – описание интерфейсов, поддерживаемых COM-сервером, которое хранится в системном реестре (рисунок )

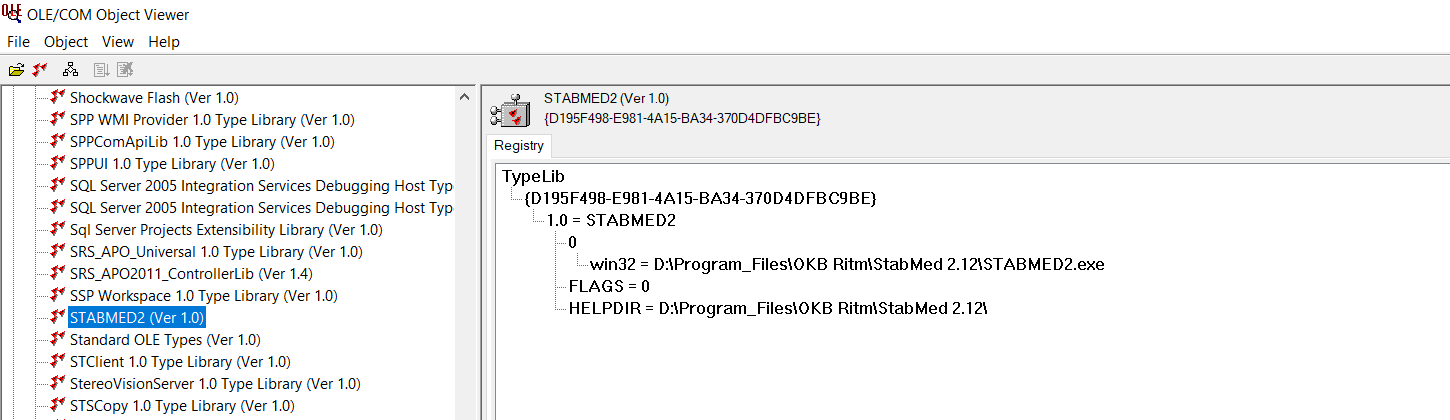


Рис. StabMed в числе других библиотек типов Component Object Model

Для этого необходимо выполнить скрипт makepy.py [6], поставляемый с вышеупомянутым пакетом pywin32, выбрав в открывшемся списке соответствующую библиотеку. После этого необходимые функции интерфейса становятся доступны Python (рис. )

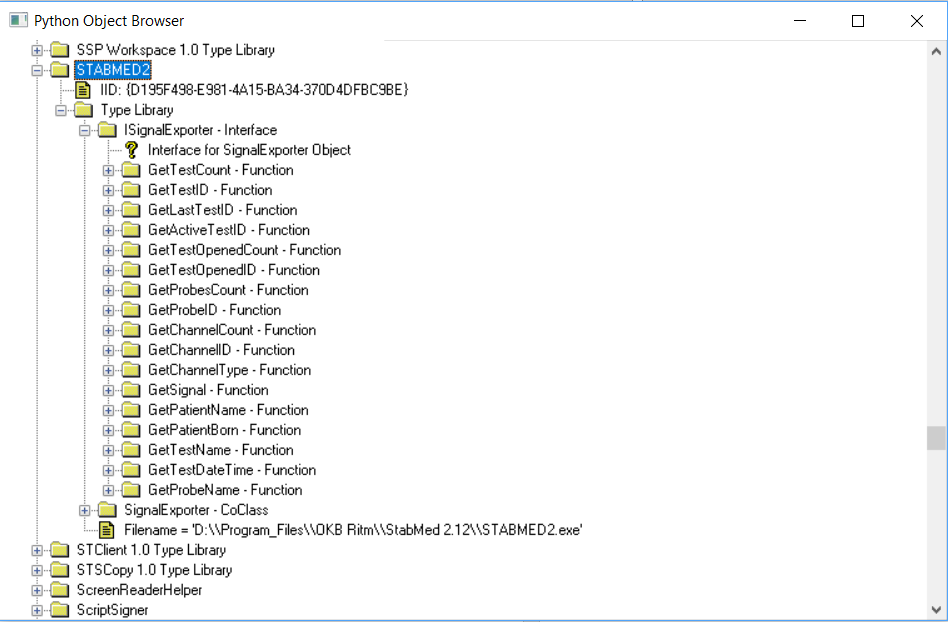


Рис. Браузер COM-объектов, доступных посредством Python

При этом автоматически генерируется описание библиотеки на языке IDL (Interface Definition Language – язык описания интерфейсов) со всеми необходимыми зависимостями. Теперь из любого скрипта на языке Python можно обращаться к COM-серверу при помощи метода win32com.client.Dispatch(), который возвращает Python-объект, имеющий все методы, описанные Приложении 1.

Разработка хранилища измерений

Для возможности дальнейшего использования полученных в процессе проведения стабилографического исследования данных, а также учитывая возможность в дальнейшем добавить в проектируемую автоматизированную систему инструменты анализа стабилограмм на базе методов машинного обучения, необходимо спроектировать хранилище получаемой системой информации.

Здесь стоит учесть, что методы стабилографии используются не только исследователями-физиологами, но и медицинскими работниками. То есть автоматизированная система по сути имеет дело с медицинской информацией. Для обеспечения удобства дальнейшего использования стабилометрических данных следует спроектировать хранилище в соответствии со стандартами хранения медицинской информации.

Таким образом, основных требований, влияющих на архитектуру, к проектируемому хранилищу данных два:

- ориентированность на дальнейшее использование методами машинного обучения;

- соответствие стандартам представления медицинской информации.

Для понимания сути первого требования необходимо понять, что представляет собой машинное обучение – таким образом, станет ясна суть требований, предъявляемых к используемым данным.

Машинное обучение (Machine Learning, ML) - это раздел информатики, изучающий алгоритмы и статистические модели, которые используются компьютерными системами для эффективного выполнения конкретной задачи без использования явных инструкций, вместо этого опираясь на шаблоны и собственные выводы. Оно рассматривается как раздел искусственного интеллекта. Алгоритмы машинного обучения строят математическую модель на основе выборочных данных, известных как «обучающие данные», для того, чтобы делать прогнозы или решения без явного программирования для выполнения задачи. Алгоритмы машинного обучения используются в широком спектре приложений, таких как фильтрация электронной почты и компьютерное зрение, где невозможно разработать алгоритм конкретных инструкций для выполнения задачи. Машинное обучение тесно связано с вычислительной статистикой, которая фокусируется на прогнозировании с использованием компьютеров. Важной частью машинного обучения является интеллектуальный анализ данных, который фокусируется на поисковом анализе данных. Применительно к бизнес-задачам машинное обучение также называется предиктивной аналитикой. Отличительной особенностью алгоритмов машинного обучения от обычных программных алгоритмов является то, что с каждой новой итерацией алгоритм выполняет возложенную на него функцию все лучше и лучше – он «учится».

В контексте пригодности сохраняемых данных для нужд машинного обучения следует рассмотреть существующие методы и алгоритмы и выявить некие общие требования, предъявляемые к исходным данным в этой сфере.

Поскольку машинное обучение является довольно обширной областью, следует определить группу задач, для решения которых будут применяться сохраняемые автоматизированной системой стабилометрические данные. На сегодняшний день можно выделить несколько проблем [14], решаемых методами машинного обучения (таблица )

Таблица Варианты применения машинного обучения с учителем, систематизированные по типам задач.

|  |  |  |
| --- | --- | --- |
| Задача | Описание | Пример применения |
| Классификация | Для каждого объекта на основе данных определяется дискретный класс | Персонализация контента, выявление производственных дефектов, изучение эффективности лекарственных средств |
| Регрессия | На основе данных предсказывается фактическое значение параметра | Прогнозы на рынке ценных бумаг, управление рисками, прогноз погоды |
| Рекомендация | Предсказывается альтернатива, которую предпочтет пользователь | Предложения продуктов, подбор персонала, предложения контента |
| Заполнение пропусков | Вывод значений отсутствующих входных данных | Неполные истории болезни, данные переписей, неполная информация о клиентах |

С другой стороны, если подходить к классификации методов машинного обучения с точки зрения принципов работы алгоритмов Machine Learning, то эти методы можно разделить на три большие группы [15]:

- методы контролируемого обучения (обучение с учителем),

- методы неконтролируемого обучения,

- методы с частичным контролем.

Во каждой группе подразумевается поиск алгоритмом закономерностей в наборе данных. Отличие состоит в том, что в случае неконтролируемого обучения алгоритм делает выводы о тех или иных закономерностях самостоятельно, генерируя, например, некую классификацию данных, а при использовании обучения с учителем и методов с частичным контролем данные необходимо предварительно классифицировать или «разметить» - этим обычно занимается человек.

В ходе интервьюирования специалистов-физиологов в процессе сбора требований к автоматизированной системе из числа приведенных в таблице задач, которые возможно решать при помощи методов Machine Learning, были отмечены классификация и регрессия. Как указывается в [14], для реализации алгоритмов данных типов необходимы предварительно «размеченные» данные. Кроме того, два типа алгоритмов Machine Learning из трех подразумевают наличие предварительно классифицированных сведений.

Таким образом, в ходе проектирования хранилища стабилометрических измерений необходимым условием является обеспечение возможности сохранения информации о принадлежности полученных данных к тому или иному классу.

Согласно [14], построение обучаемой модели проходит в несколько этапов, первым из которых является сбор и подготовка данных. Поскольку в рамках данной работы проектируется не сама классифицирующая модель, а лишь хранилище для данных, необходимо лишь частично воспроизвести первый этап и те его шаги, которые непосредственно повлияют на структуру хранилища.

Этап сбора и подготовки данных состоит из нескольких шагов:

1. Определение набора входных признаков
2. Подготовка данных к моделированию
3. Визуализация данных

Последний шаг будет, по сути, реализован в ходе проектирования и реализации модуля создаваемой автоматизированной системы, отвечающем за визуализацию полученных и обработанных данных, на структуру хранилища стабилографических данных повлияют только первые два.

Для определения набора входных признаков следует обратиться к ранее собранным требованиям к автоматизированной системе обработки результатов стабилографических измерений (см. главу 1), рассмотрев их с точки зрения требований к набору данных для применения в алгоритмах Machine Learning. В отношении подготовки стабилографических данных к использованию в ML-алгоритмах, имеет значение требование о маркировании сигнала как полученного в результате обследования пациента с диагностированной патологией и как сигнала, по предположению специалиста свидетельствующего о наличии патологии. Принимая во внимание описанный выше алгоритм проведения стабилографического исследования, одним из главных вопросов, на который с большой долей вероятности должна будет ответить будущая модель, является «есть ли у данного пациента какое-либо заболевание».

Далее, в соответствии с вышеописанным алгоритмом проектирования ML-модели, определим набор входных признаков, на основе которых можно ответить на данный вопрос. В ходе анализа предметной области (см. главу 1), а также в процессе обсуждения с экспертами-физиологами были выявлены следующие репрезентативные стабилографические показатели, которые могут свидетельствовать о наличии того или иного заболевания:

- наличие диапазонов частот, мощность которых составляет больше 60%

- относительно высокие значения дисперсии для координаты центра давления

- заключение специалиста о возможном наличии патологий;

- статус здоровья обследуемого пациента.

Преимуществом данных признаков является то, что они представлены вещественным типом данных – в дальнейшем это может значительно облегчить их использование в машинном обучении [14].

Здесь может возникнуть вопрос о целесообразности хранения прямых измерений и рассчитываемых на их основе косвенных показателей, ведь с точки зрения использования памяти было бы логичнее хранить исключительно прямые измерения, а остальное считать по необходимости. Однако здесь логично предположить частую необходимость обращения к расчетным параметрам и принять во внимание тот факт, что процесс «обучения» модели в машинном обучении сам по себе занимает довольно длительное время. При этом «учиться» будет с большой долей вероятности будет на базе косвенных показателей, поскольку следует разграничивать задачи детектирования аномалий в сигналах методами машинного обучения и классификацию объектов на основе конечного числа параметров. В связи с этим, а также учитывая относительно малое количество расчетных параметров хранение этих параметров в данном случае оправдано.

Целевой переменной в данном случае следует объявить маркер «наличие патологии». Этот показатель характеризует набор данных как полученный от пациента, наличие заболевания у которого было выявлено до получения проведения стабилографического исследования и, соответственно, он хранит ответ на вопрос о том, болен ли пациент.

Подготовка стабилографических данных к моделированию будет заключаться лишь в преобразовании категориальных показателей в числовые. В наборе показателей два категориальных признака: заключение специалиста о возможном наличии патологии и статус здоровья пациента. В соответствии с требованиями к автоматизированной системе они могут принимать значение «здоров» и «болен» - данное множество значений естественно представить нулем и единицей соответственно.

Окончательно множество признаков, которые могут понадобится в рамках использования хранимых стабилографических данных для целей машинного обучения, представлен в таблице .

Далее следует перейти ко второму требованию, непосредственно связанному с хранилищем стабилометрических данных - соответствие стандартам представления медицинской информации. На сегодняшний день существует целый ряд стандартов подобного рода. Таким образом, перед непосредственно проектированием и реализацией хранилища данных для автоматизированной системы следует выбрать стандарт, на котором будет базироваться хранилище. Ниже проводятся краткие обзор и сравнительный анализ различных стандартов, касающихся медицинских данных.

Компьютеризация здравоохранения и массовое внедрение информационных систем в медицинских учреждениях привели к необходимости создания стандарта хранения и передачи медицинских данных.

Одним из таких стандартов является Health Level 7 или HL7. Это стандарт для обмена, управления и интеграции клинических данных в электронном виде. HL7 спроектирован по аналогии с семью уровнями эталонной модели взаимодействия открытых систем OSI (рис. ) и поддерживает выполнение таких функций, как:

- структурирование передаваемой информации;

- проектирование медицинских информационных систем;

- согласование форматов клинических данных при их передаче;

- безопасность;

- идентификация пациентов и медицинского персонала.



Рис. Структура модели Health Level 7

HL7 сейчас представляет собой три поколения стандартов. Первая версия стандарта HL7 V1 предназначена только для апробации модели и уточнения содержания и структуры данных. По факту же массовое использование стандарта началось после релиза версии HL7 V2. Данная версия содержит описание обмена административной, медицинской и финансовой информацией в виде текстовых сообщений. Более поздняя версия HL7 2.5 является наиболее распространенной. Наибольшее число сложностей связано с единой семантикой модели данных. Поля и отношения добавляются в систему по мере возникновения у пользователей необходимости в них; отсутствует механизм расширения и модернизации структур данных без необходимости выпуска новой версии стандарта.

HL7 V3 – последняя версия стандарта – не является обратно совместимой с версией V2. Это семейство стандартов, основанных на эталонной информационной модели, определенных структурах и типах данных, словарях терминов, характерных для предметной области, и строго определенной методике разработки стандартов. Однако это сделало стандарт чрезвычайно сложным для реализации, в связи с чем данная версия не получила широкого распространения. Актуальной версией Health Level 7 на текущий момент является HL7 FHIR - ресурсы для обмена медицинской информацией. На сегодняшний день эта версия находится на этапе проекта для пробного использования, однако ряд крупных компаний уже используют HL7 FHIR при разработке своих платформ.

Диагностика и лечение пациентов подразумевает генерирование и хранение больших объемов текстовой и графической информации (информационный объем рентгеновских снимков и результатов томографии может составлять несколько десятков гигабайт). В связи с развитием технологий и увеличением разрешения светочувствительных матриц, объем медицинской цифровой графики лавинообразно растет, при этом задача ее унификации становится все актуальнее. В связи с этим еще в 80-х годах прошлого века был разработан стандарт DICOM (Digital Imaging and Communications in Medicine). Данный стандарт формализует создание, хранение, передачу и отображение изображений и документов (рис. ). Для эффективной работы необходимо комплексное решение по управлению всеми диагностическими данными пациента, начиная вводом изображения в систему и заканчивая его архивацией в конце лечения. Эти задачи легли в основу открытой архитектуры стандарта DICOM, что позволило организовать не только их передачу, но и автоматизированную обработку данных, что сокращает время на подготовку к диагностике либо лечению и ускоряет проведение исследований. Для увеличения эффективности DICOM может использоваться на всех этапах диагностики, снижает затраты, сокращая время подготовки к обследованию, отбраковывая некачественные данные и уменьшая потери качества изображения.

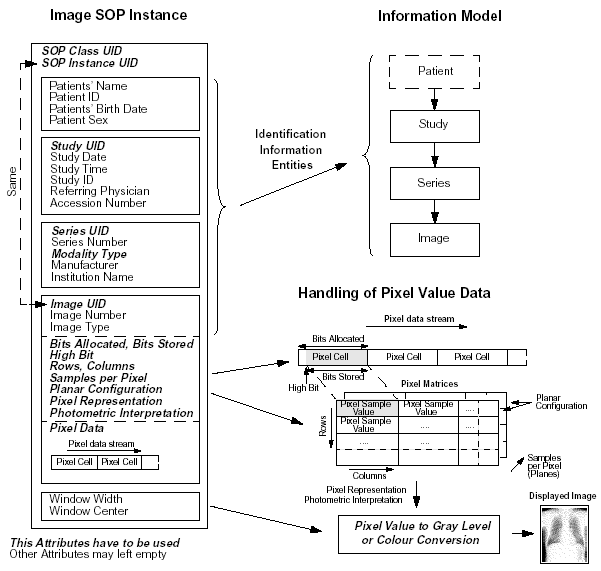


Рис. Структура данных, реализуемая DICOM (изображение взято с официального сайта стандарта)

Со временем количество стандартов, как и их качество и характеристики медицинского оборудования, возросло. Это вызвало проблемы с совместимостью различных медицинских систем и их интеграцией. Результатом решения этих проблем стала система стандартов XDS (Cross-Enterprise Document Sharing) - стандарт каталогизации обмена медицинской информацией между организациями и системами.

Важным следствием внедрения и совершенствования этих стандартов явился переход к стандартизированной модели хранения медицинских данных. Сегодня вся информация может храниться отдельно, при этом после завершения жизненного цикла клинических данных нет необходимости осуществлять дорогостоящий и долгий процесс перевода их из одного формата в другой. Такой подход позволяет создавать унифицированные базы данных, доступ к которым имеют любые медицинские системы в разных регионах и даже странах.

На основе всего вышеизложенного при проектировании автоматизированной системы следует воспользоваться спецификациями DICOM, как стандарта, изначально разработанного для хранения медицинских данных. В пользу выбора именно этого стандарта говорит также отсутствие избыточности в сравнении с HL7, что отмечается в том числе в [17].

На основе поставленной задачи и выявленных требований, в соответствии со стандартом DICOM [18] и описанной в [19] процедурой проектирования следует составить пул сущностей базы данных, которая будет являться хранилищем стабилографических измерений (табл. ).

Таблица Описание пула сущностей

|  |  |  |
| --- | --- | --- |
| Номер сущности | Имя сущности | Определение сущности |
| 1 | Пациент | Человек, проходящий обследование |
| 2 | Исследование | Совокупность результатов цикла измерения |
| 3 | Серия | Совокупность сигналов, одновременно считанных с разных датчиков |
| 4 | Сигнал | Множество точек в координатной плоскости «время-координата» |

В соответствии с DICOM матрица связей имеет довольно тривиальный вид, представленный таблицей

Таблица Матрица связей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | E1 | E2 | E3 | E4 | E5 |
| E1 |  | X |  |  |  |
| E2 | X |  | X |  |  |
| E3 |  | X |  | X |  |
| E4 |  |  | X |  | X |
| E5 |  |  |  | X |  |

Каждая связь относится к типу «один-ко-многим» и каждой из них следующая сущность является дочерней по отношению к предыдущей. Такой подход, продиктованный стандартом DICOM, обеспечивает простоту реализации и легкость в понимании отношений между сущностями. При этом данная реализация полностью соответствует бизнес-логике предметной области: обратившись к описанному выше алгоритму проведения стабилографического исследования, можно убедиться, что один пациент может принимать участие в нескольких исследованиях, однако исследование подразумевает участие только одного пациента. В свою очередь исследование включает в себя серию измерений, производимых специалистом. А измерение, в силу специфики аппаратно-программного комплекса «Многофункциональное кресло»

Построенная на основе приведенных выше пула сущностей и таблицы описания связей модель уровня сущностей в виде ER-диаграммы представлена на рисунке

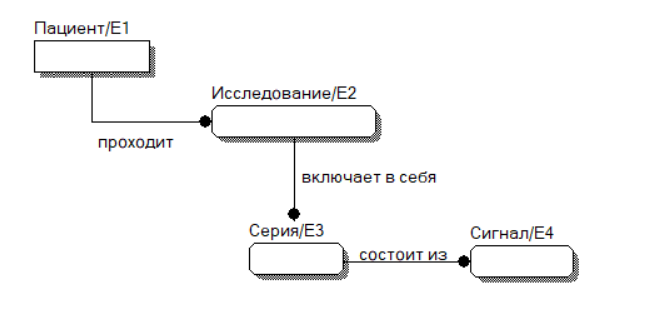


Рис. ER-диаграмма базы данных

В соответствии с параметрами, необходимыми для построения моделей машинного обучения, о чем говорилось выше, перечислим атрибуты сущностей (табл. )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя и номер сущности | Имя атрибута | Имя домена | Признак обязательности | Примечание |
| Пациент/Е1 | ID |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Исследование/Е2 | StudyID |  |  |  |
|  | AccessionNumber |  |  |  |
|  | Date |  |  |  |
|  | Time |  |  |  |
|  | Description |  |  |  |
| Серия/Е3 | SeriesID |  |  |  |
|  | Description |  |  |  |
|  | Date |  |  |  |
|  | Time |  |  |  |
|  | Number |  |  |  |
|  | Modality |  |  |  |
|  | Region |  |  |  |
| Сигнал/Е4 |  |  |  |  |

Тогда полноатрибутная диаграмма модели будет выглядеть следующим образом (рис. ):

Скрипт на языке SQL, который при выполнении создаст базу данных, приведен в Приложении 3.

Далее встает вопрос о реализации базы данных. Сотрудниками МПГУ в рамках работы с АПК «Многофункциональное кресло» используются персональные компьютеры с относительно малой мощностью и ограниченным количеством памяти. На многих из них установлены морально устаревшие версии операционной системы Windows, а сами сотрудники не обладают достаточными знаниями для самостоятельной конфигурации своих электронно-вычислительных машин. В связи с этим использование таких СУБД, как MySQL, Firebird или PostreSQL не является целесообразным. Для задач, решаемых проектируемой автоматизированной системой, а именно для стандартизированного хранения довольно тривиальной по своей структуре базы данных, будет достаточно встраиваемой СУБД. Учитывая реализацию системы на Python, предпочтение следует отдать поддерживающей данный язык программирования СУБД SQLite. Здесь же стоит отметить наличие для данной системы управления базами данных небольших по объему и простых по своей структуре кроссплатформенных приложений, которые позволяют взаимодействовать с базами данных под управлением SQLite без запуска «носителя».

Программный модуль хранилища базы данных в соответствии с предъявляемыми требованиями и алгоритмом осуществления стабилометрического исследования должен поддерживать следующий функционал:

- создание базы данных на персональном компьютере, на котором функционирует автоматизированная система (необходимо в начале работы с автоматизированной системой);

- добавление в базу данных информации об очередном стабилометрическом исследовании (ключевой процесс, отвечающий за накопление стабилометрических данных);

- экспорт информации о стабилометрических исследованиях, хранящихся в базе данных на текущий момент (может потребоваться для отладки и в дальнейшем при использовании сохраненных данных для построения модели машинного обучения).

Реализация данных функций на языке Python в соответствии со спецификациями SQLite [20] приведена в Приложении 3.

Разработка модуля математической обработки сигналов

Для проведения частотного анализа стабилометрических сигналов необходимо совершить перевод множества полученных значений из координатной системы «время – значение координаты», в которой пользователь получает стабилометрический сигнал, в координатную систему «частота – амплитуда». В ходе интервьюирования сотрудников кафедры анатомии и физиологии человека и животных МПГУ, а также изучения материалов на тему анализа стабилометрических данных [5] было установлено, что в качестве инструмента для преобразования стабилометрического сигнала с целью его дальнейшего частотного анализа следует использовать быстрое преобразование Фурье.

Данная математическая операция реализована в библиотеке numpy. Функция numpy.fft() может принимать в качестве аргумента как numpy-массивы, так и стандартные списки Python.

Помимо преобразований Фурье, с полученным сигналом, в соответствии с теорией (см. главу 1), следует произвести ещё ряд математических операций. Необходимо рассчитать такие параметры, как:

- дисперсия сигнала;

- мощности частот сигнала;

Разработка графического пользовательского интерфейса

Среди требований, предъявленных к разрабатываемой системе, присутствуют требования, непосредственно касающиеся графического пользовательского интерфейса. Для проектирования графического пользовательского интерфейса был выбран язык Python версии 3.7 вкупе c библиотекой PyQt. В рамках данной работы использовалась версия PyQt5.

PyQt - это набор привязок Python v2 и v3 для инфраструктуры приложений Qt компании Qt. Он работает на всех платформах, поддерживаемых Qt, включая Windows. PyQt5 поддерживает Qt v5. Привязки реализованы в виде набора модулей Python и содержат более 1000 классов.

Данный инструмент был выбран по причине того, что PyQt объединяет кроссплатформенную среду приложений Qt C ++ и кроссплатформенный интерпретируемый язык Python. Он включает в себя абстракции сетевых сокетов, потоков, Unicode, регулярных выражений, баз данных SQL, SVG, OpenGL, XML, полнофункциональный веб-браузер, справочную систему, мультимедийную среду, а также богатую коллекцию графических виджетов, что дает широкие возможности по дальнейшему расширению функционала разрабатываемой автоматизированной системы.

Классы Qt используют механизм сигнал / слот для связи между объектами, который является безопасным по типу, но слабо связанным, что облегчает создание программных компонентов многократного использования.

Qt также включает Qt Designer, дизайнер графического пользовательского интерфейса. PyQt может генерировать код Python из Qt Designer. Также возможно добавить новые элементы управления GUI, написанные на Python непосредственно разработчиком, предоставляя большие возможности для кастомизации разрабатываемого интерфейса.

Для понимания структуры программной реализации графического пользовательского интерфейса следует вкратце рассмотреть структуру библиотеки PyQt и механизмы работы классов и методов.

Иерархия классов в PyQt представлена на рис..

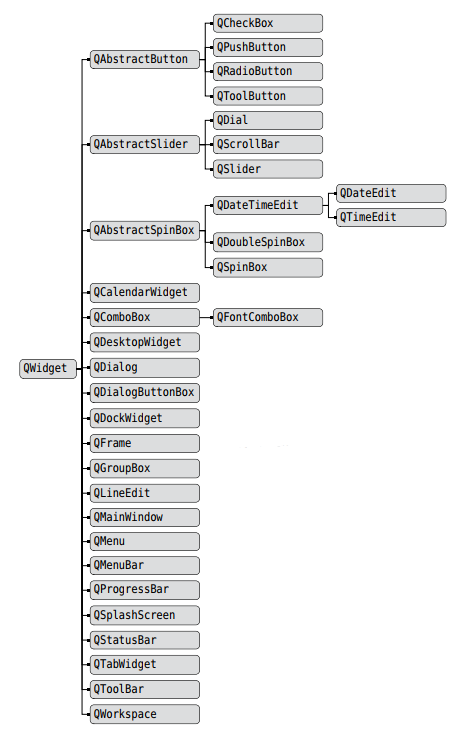


Рис. Иерархия классов в PyQt5

QWidget является базовым классом, от которого наследуются классы-элементы графического интерфейса, в названиях которых содержится прямая отсылка к их внешнему виду и функционалу. Так, класс QCheckBox представляет возможность добавлять в графический интерфейс пользователя чек-боксы, а QMenuBar отвечает за отображение в верхней части окна пунктов «File», «Edit» и так далее.

Взаимодействие пользователя с окном осуществляется, как было сказано выше, через механизм сигналов и событий. Событие – это своего рода извещения о выполнении пользователем какого-либо действия либо возникновение некоторого условия в самой системе. В ответ на события система генерирует определенные сигналы, которые являются представлением системных событий внутри PyQt. При помощи метода connect() класса QObject тем или иным событиям ставится в соответствие функция или метод, то есть событию назначается обработчик, который будет вызываться при обнаружении сигнала.