Проект разработки информационной системы EDDY.

16 апреля 2015 г.

1 Этапы работ.

1. Сбор требований о системе (функционал, нагрузка, окружение).
2. Подбор технических средств (в т.ч. и программных), необходимых для решения задачи.
3. Построение модели архитектуры системы с высоком уровнем абстракции.
4. Оценка сроков выполнение отдельных этапов и «фич» с учетом подобранных технологий и архитектуры.
5. Повторное обсуждение с заказчиком:
   - Выделение наиболее приоритетных требований.
   - Построение backlog — все, что должно быть сделано до конца текущего финансового года.
   - Построение «списка желаний» — улучшения системы, которые было бы неплохо сделать, но в backlog они не попали из-за нехватки времени или малой значимости.
   - Определение методов контроля процесса разработки заказчиком и тестирования системы (например, внутренний заказчик).
   - Сбор дополнительный сведений (например, дополнительные затраты, покупка технических средств и др.).
6. Анализ backlog:
   - Декомпозиция backlog на отдельные задачи.
   - Выявление блоков задач, которые необходимо выполнять последовательно.
   - Проверка оценок срока выполнения.
7. Корректирование набора технических средств с учетом пожеланий и возможностей заказчика.
8. Более детализированная проработка архитектуры.
9. Финальная встреча с заказчиком: в случае подтверждения полученных в процессе предыдущих встреч данных, переходим к разработке. В противном случае, возвращаемся к пункту 5.

Разработка проходит итерационным методом, каждая итерация выглядит следующим образом:

1. Выделение задач, которые войдут в текущую итерацию разработки.
2. Набросок нескольких последующих итераций.

3. Активная стадия итерации:

    - Написание программного кода.
    - Покрытие кода тестами.
    - Совещания, обсуждения (brainstorming) по текущим вопросам (например, улучшение качества процесса разработки, дополнительная информация от заказчика, проблемы с текущими задачами и др.)

4. Если текущая итерация считается вехой (milestone), то происходит следующее:

    - Интеграционное, системное, нагрузочное тестирование.
    - Демонстрация заказчику.
    - Тестирование и фиксация результатов.

5. Переход к следующей итерации.

Предполагается, что в процессе разработки используются принципы «непрерывной интеграции» (continuous integration), поэтому процесс тестирования можно считать непрерывным и не зависящим от текущей стадии итерации.

В конце текущего финансового года:

1. Процесс приемки заказчиком результатов.
2. Финальное тестирование (интеграционное, системное, нагрузочное).
3. Если система и ее окружение (необходимые со-системы, «железо») готовы для ввода в эксплуатацию, происходит ввод в эксплуатацию.
4. Возвращаемся к формированию backlog и «списка желаний» на следующий финансовый год.

2 Требования к системе

2.1 Архитектурные замечания

В системе не выделяются пользователи (users) в классическом понимании этого термина. Считается, что система является частью одной большой системы, в которой присутствуют следующие компоненты:

- Система управления пользователями: оперирует набором разрешений и ролей (групп разрешений) и всегда способна корректно ответить на запрос пользователя (в простейшем случае — разрешить или запретить действие).
- Клиентский сервер: работает непосредственно с клиентским кодом (HTML, js).

Используя эти компоненты, механизм работы объемлющей системы можно описать следующим образом: принцип работы клиентского сервера назовем механизмом подписок: любое действие пользователя является некоторой подпиской (например, подписка на просмотр данных какого-то канала, подписка на изменение настроек вычислений и т.д.). Подписки бывают статическими и динамическими. Для ответа на статическую подписку достаточно ответить один единственный раз, в то время как для ответа на динамическую подписку необходимо посылать данные в режиме on-line до тех пор, пока подписка не будет отменена. Всякая подписка содержит в себе ключ авторизации, по которому система контроля пользователями способна корректным образом разрешить или

запретить выполнение подписки. В случае запрета, клиентский сервер сообщает пользователю о том, что у него недостаточно прав для совершения запрошенного действия. В противном случае, совершается комплекс действий, который приводит к выполнению требуемого действия.
Заметим, что помимо этих компонент в объемлющей системе могут присутствовать и другие, однако, они никоим образом не связаны с работой EDDY.
Особо стоит выделить еще один программный продукт, который нельзя отнести ни к объемлющей системе, ни к системе EDDY. Речь идет о программе-адаптере, которая способна считать данные из источника данных и направить их на вход EDDY. Адаптер не является частью рассматриваемой системы так как должен разрабатываться индивидуально для каждого заказчика в соответствии с имеющимся оборудованием и окружением. Считается, что между адаптером и рассматриваемой системой существует некоторая прослойка (связующая среда), которая работает независимо от обоих участников. Единственной целью этой прослойки является передача данных от адаптера к EDDY. Роль такой прослойки может играть, например, распределенная очередь сообщений (Kafka, Kestrel, ApacheMQ, RabbitMQ). Сам же адаптер в этом случае работает по принципу проталкивания данных (push), то есть независимо от состояния EDDY, помещает данные в коммуникативную среду. Ответственность за доставку и обработку данных сообщений в этом случае ложится на прослойку и, возможно, другие системы (например, простую систему, которая сохраняет все помещенные в прослойку данные в базу данных). В описываемой архитектуре адаптер нельзя считать пользователем системы, поскольку его работа никак не зависит от рассматриваемой системы. Более того, адаптер не обязан ничего знать о существовании чего-то, работающего с прослойкой.
Таким образом, у рассматриваемой системы имеется всего один пользователь — клиентский сервер.

2.2 Бизнес требования

- Обработка входящих данных в соответствии со спецификациями, настраиваемыми пользователями.
- Динамическая смена настроек вычислений.

2.3 Пользовательские требования

- Для любого доступного канала система должна поддерживать следующие типы динамических подписок:
  - График спектра.
  - Средние значение за конфигуриремый промежуток времени.
  - Максимальные значения за всю историю.
  - Историю спектра (campbell plot).
  - Значения амплитуды, частоты и фазы пиков, а также среднее квадратическое по заданным диапазонам частот спектра.
  - Исторические значения амплитуды, частоты и фазы пиков, а также среднее квадратическое по заданным диапазонам частот спектра.
- Для любого доступного канала система должна поддерживать следующие типы статических подписок:
  - История спектра за конкретный период времени (campbell plot).
  - Исторические значения амплитуды, частоты и фазы пиков, а также среднее квадратическое по заданным диапазонам частот спектра за заданный промежуток времени.
- Добавить/удалить вычисление для канала.

  - В случае, когда удаляются все вычисления для канала, система сама помещает одно стандартное (например, с минимальной для системы нагрузкой) вычисление для канала.

- Возможность конфигурировать следующие параметры вычисления:
  - Уменьшение частоты дискретизации:
    * Путем выборки каждого n-го наблюдения.
    * Путем усреднения по окну конфигурируемой длины (с использованием интерполяции, если необходимо).
    * Путем выборки значений, соответствующих одинаковым угловым изменениям вращающегося элемента (для сигналов, которые измеряют величины, связанные с вращением некоторого элемента).
  - Частотная фильтрация выборки, полученной после уменьшения частоты дискретизации.
  - Разбиение на части блоков данных, полученных после уменьшения частоты дискретизации и/или фильтрации, используемых как вход для подсчета FFT (Fast Fourier Transform)
  - Весовую функцию (window function), используемую для улучшения качества работы FFT.
  - Парметры для диапазонов частот:
    * Границы диапазона.
    * Количество пиков, которые необходимо найти в каждом диапазоне.
  - Период времени, по которому будет браться усреднение амплитуд спектров.

2.4 Функциональные требования

Для удовлетворяения пользовательских требований, функционал системы должен обладать следующими возможностями:

- Получение данных.
- Дубликация значений каналов по числу вычислений, соответствующих каналу.
- Уменьшение частоты дискретизации.
- Частотная фильтрация.
- Разбиение данных на блоки произвольного размера и перекрытия.
- Вычисление заданного числа точек весовой функции.
- Быстрое преобразование Фурье.
- Поиск произвольного количества пиков в спектре для заданного диапазона частот.
- Усреднение амплитуд пиков по заданному временному промежутку.
- Обновление максимальных значений спектра.
- Отсылка спектров, информации о частотных диапазонах, усредненных и максимальных амплитудах на клиентский сервер.

2.5 Производительность

Система должна обрабатывать данные как можно быстрее, поскольку конечному пользователю важно видеть актуальные данные. Предположим, что в систему поступают k блоков данных за секунду. Постулируется, что изменение графиков чаще, чем раз в секунду не будет улавливаться человеческим глазом. В таком случае, все k блоков буферизуются и раз в секунду засылаются в систему одним большим блоком. При такой организации система должна обрабатывать запрос не дольше секунды, потому что в противном случае буфер на входе в систему рано или поздно переполнится, что приведет к краху системы.
Нагрузка на систему предполагается следующая: 300 каналов, вещающих с частой 25kHz, по $3--4$ вычисления на каждый.

2.6 Требуемая инфраструктура

Системе для полноценного функционирования требуются следующие компоненты:

- Выделенный сервер/кластер с настроенным окружением (например, Storm/Spark) для запуска системы.
- Сервер, назначением которого является обслуживание коммуникативной среды для пересылки данных из адаптера в систему и из системы на клиентский сервер.
- Сервер со следующими вспомогательными службами:
  - Клиентский сервер (например REST на основе Jersey).
  - Система управления пользователями.
  - Сервер реляционной базы данных (например, бесплатный MySQL) для хранения редко меняющейся информации (например, данных о пользователях).
- Выделенный сервер/кластер noSQL базы данных для хранения результатов вычислений.