**ГЛАВА 3 ПРОГРАММОЕ ОБЕСПЕЧЕНИЕ**

**3.1 Требования к программному обеспечению**

**3.2 Выбор программной платформы**

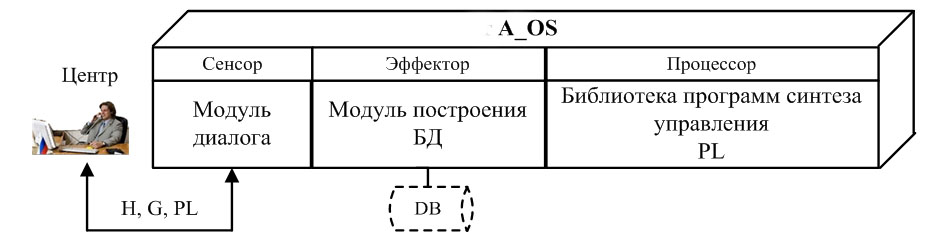
Правильный выбор программной платформы, адекватной свойствам и целям решаемой задачи, является важнейшей проблемой разработки компьютерных систем, т.к. от него зависит успех всего проекта. Неправильный выбор приводит к удорожанию проекта и усложнению поддержки жизненного цикла целевой системы. От выбора платформы так же зависит удобство и время отладки кодов программ, возможность использования ранее разработанных фрагментов программ, качество сервисной поддержки, трудоемкость документирования проекта и т.д. Поэтому вопрос выбора платформы для реализации разработанных в процессе дипломного проектирования моделей и алгоритмов требует отдельного рассмотрения.

Сюда обзор ОС, Xcode, Objective-C, Cocoa

**3.3 Архитектура системы**

Архитектура системы включает модули, реализующие процессы решения задачи. Выше показано, что решение необходимо свести к пяти процессам, обеспечивающих в комплексе выполнение мониторинга, соответственно агентов должно быть пять.

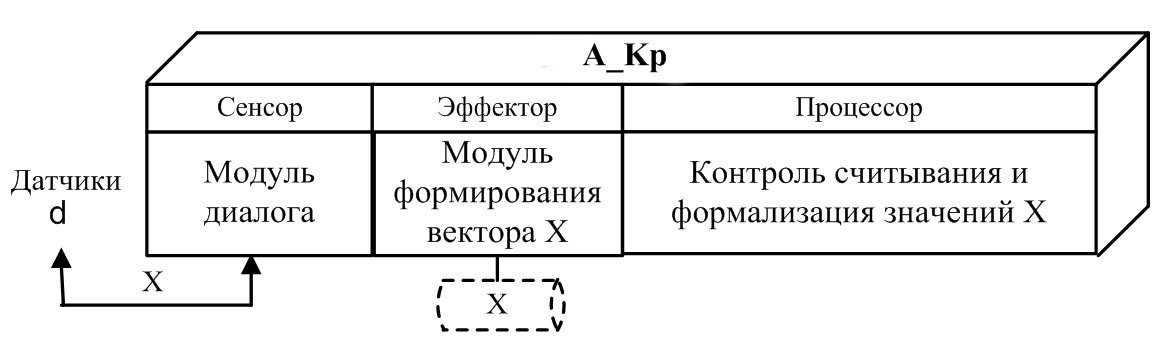
Синхронизация процессов осуществляется руководством организации, поэтому в первую очередь построим архитектуру модуля, формирующего модель иерархии и включающий атрибуты всех одушевленных и виртуальных (программы, устройства) участников организации(Рисунок 3.1).

****

**Рисунок 3.1** – **Архитектура модуля построения модели организации**

Сенсорный элемент модуля A\_OS в диалоге в руководством организации получает всю необходимую информацию как о самом центре, так и об объектах наблюдения. Вся полученная информация аккумулируется в базе данных (DB). Программы анализа данных находятся в библиотеке PL, которая формируется на стадии компиляции агента.

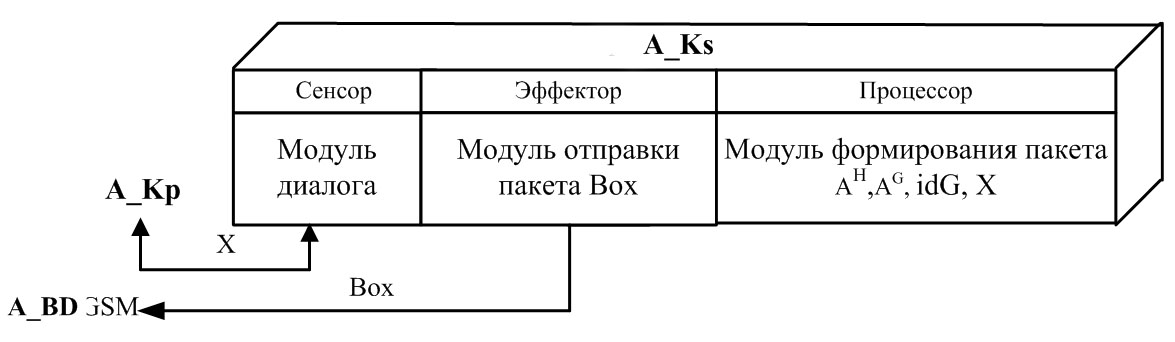
Первичным источником информации в мониторинге данного типа являются датчики. Соответственно, второй модуль должен получить значения признаков, формализовать их и передать агенту связи. Соответствующая архитектура представлена на рис.3.2.



**Рисунок 3.2** – **Архитектура модуля для формирования вектора значений показателей**

Работа модуля агента осуществляется, как правило, в дискретном режиме, т.е. значения показателей снимаются через определенный промежуток времени, установленный экспертом при разработке контроллера. Аппаратно модуль представлен комплексом датчиков и программируемым контроллером. Датчики измеряют значения <X> параметров X, контроллер формализует их и проводит первичную статистическую обработку.

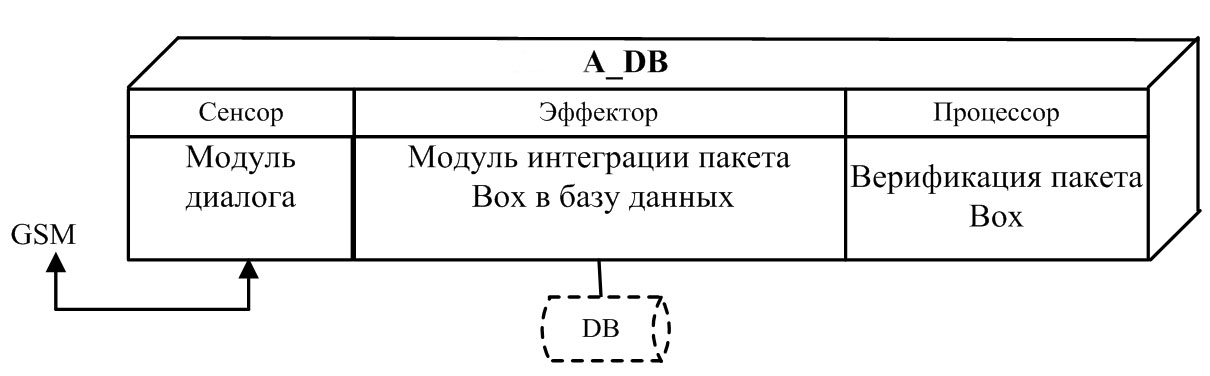
Далее активизируется модуль, который получает вектор X и формирует пакет-сообщение Box о текущем состоянии объекта для отправки в центр. Пакет содержит вектор показателей и идентификационные атрибуты объекта. Архитектура модуля A\_Ks, выполняющего эти операции, представлена на рис. 3.3.

** Рисунок 3.3** – **Архитектура модуля для формирования пакета**

Завершив формирование пакета, модуль A\_Ks выходит в сеть GSM по протоколу GPRS и отправляет сообщение Box на сервер в центр.

Аппаратно модуль A\_Ks представлен программируемым контроллером и средствами сопряжения с внешними коммуникациями.

Функции получения и интеграции пакета в базу данных DB выполняет агент A\_DB, архитектура которого представлена на рисунке 3.4.

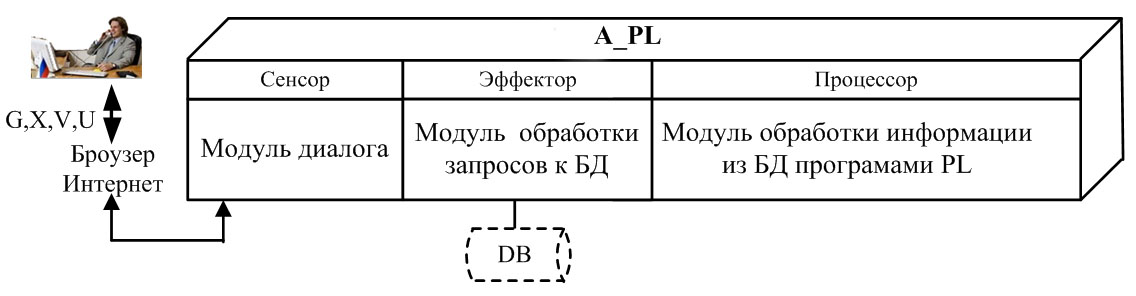
****

**Рисунок 3.4** – **Архитектура модуля для решения прикладных задач**

Агент анализирует входную информацию, идентифицирует задачу и объект, проверяет корректность значений вектора X, выполняет статистическую обработку, интегрирует результат в базу данных центра.

Информация, интегрированная в базу данных, может использоваться для решения задач синтеза состояния объектов, выработки управления, прогноза и т.д. Кроме того, она может использоваться в OLAP-анализе методами Data Mining и Knowledge Discovery. Доступ к базе осуществляется посредством абонентских диспетчерских программ, совместимых с интернет-браузерами MS Explorer, Opera, FireFox и др.

Обработку запросов пользователей к базе данных осуществляет агент A\_PL, архитектура которого представлена на рис. 3.5.

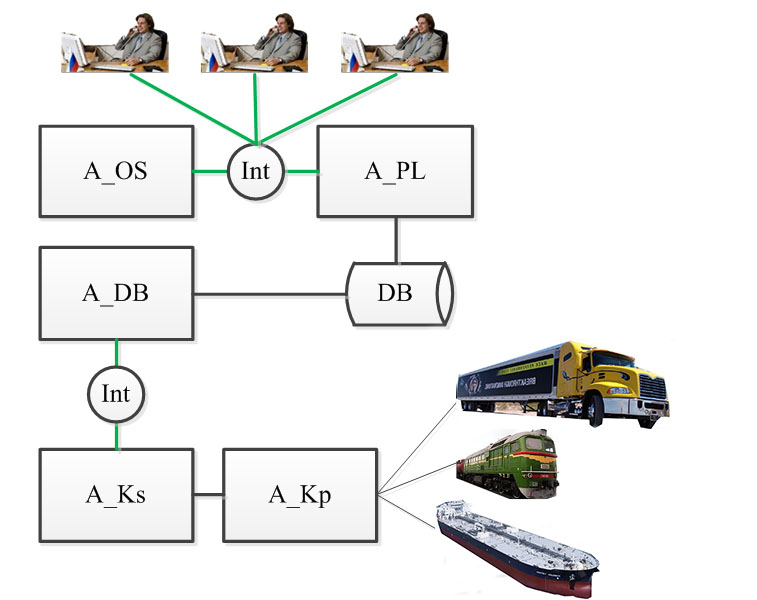
****

**Рисунок 3.5** – **Архитектура модуля обработки запросов и решения прикладных задач**

Доступ к ресурсам сервера (базе данных и пакету программ обработки) осуществляется после проверки входного кода пользователя. Использование общедоступных браузеров позволяет контролировать состояние удаленных объектов с помощью стационарных и портативных компьютеров, коммуникаторов, планшетов и т.д. Библиотека PL имеет модульную структуру, поэтому список решаемых задач постоянно расширяется.

Таким образом, архитектура системы мониторинга можно представить как совокупность пяти программных агентов, сервера, базы данных и стандартных коммуникационно-диалоговых средств Интернет. Такой подход позволяет модернизировать аппаратную и программную части каждого из участников мониторинга, не затрагивая остальных. При изменении требований коммуникаций достаточно перепрограммировать контроллер агента A\_Ks.

Пример построения архитектуры на основе разработанного подхода для удаленного мониторинга автопоездов, тепловозов, танкеров представлен на рис.3.6.



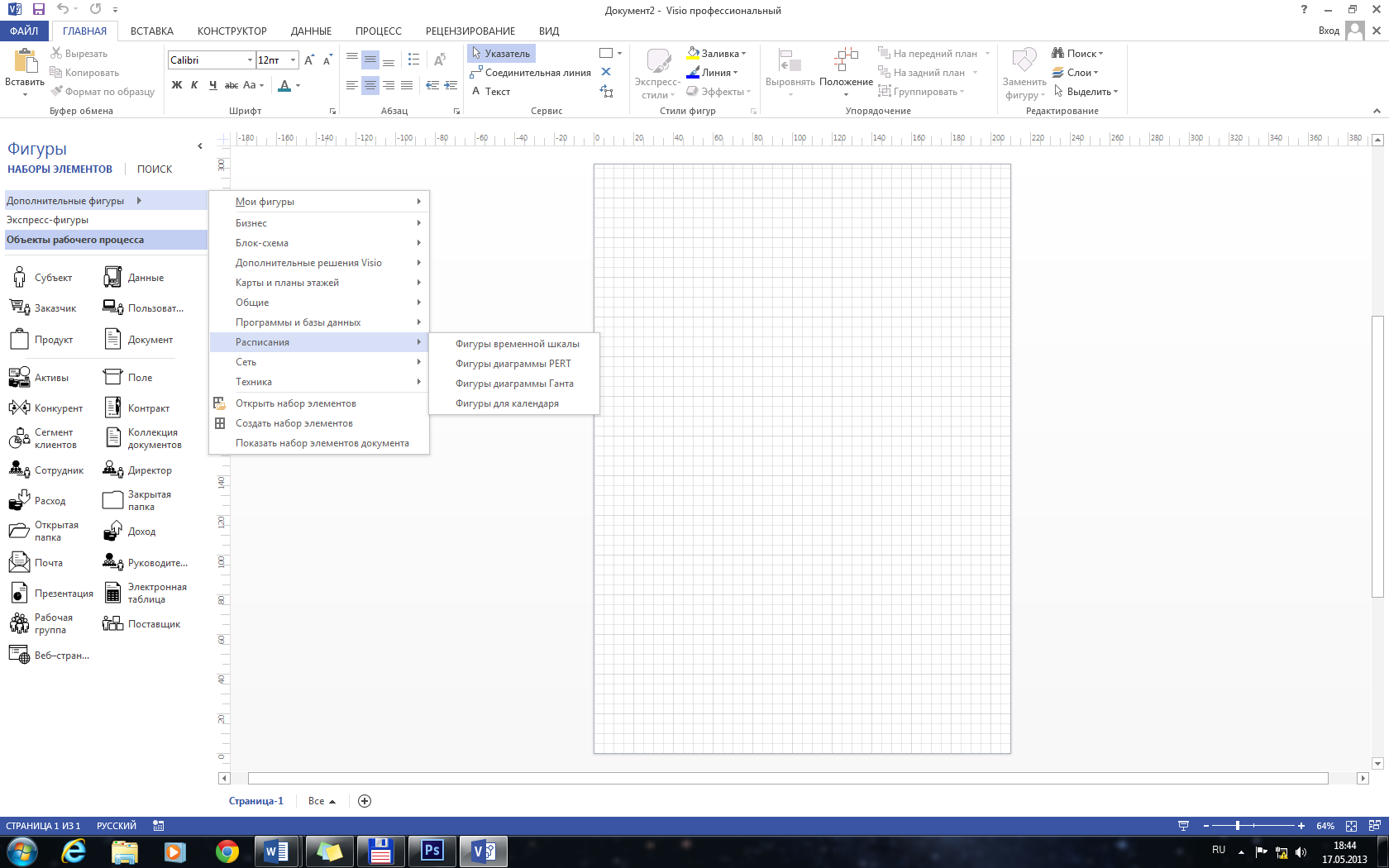
**Рисунок 3.6** – **Типовая архитектура системы мониторинга**

Для практической реализации аналогичных архитектур разработан широкий спектр датчиков и программируемых контроллеров. В настоящее время датчики могут снимать в среднем около 20 различных параметров (температуру, вибрацию, давление, уровень топлива в баках, обороты двигателей, токи, местоположение абонента в GPS, ГЛОНАСС и т.д.), характерных для мобильных механизмов (тепловозов, вагонов, тягачей, грузовиков, танкеров и т.п.). Возможность программирования обеспечивает быструю настройку контроллеров на новую предметную область и выбор конфигурации датчиков.

Основной нерешенной проблемой, повышающей стоимость данного варианта архитектуры, является большой объем ручного труда при настройке программ обработки библиотеки PL на конкретную предметную область. При этом количество продукций при логическом выводе состояния V объектов G в зависимости от значений X, и синтез управления U в зависимости от V может доcтигать нескольких тысяч, что затрудняет не только отладку, но и понимание логики массива продукций неподготовленным пользователем.

**3.3 Эскизы интерфейсов**

Разработка эскизов интерфейсов является трудоемкой операцией, которая обычно выполняется программистом или дизайнером вручную. Для максимального упрощения рисования часто достаточно сложных схем используется специализированная компьютерная система Visio 2013, которая с 2000 года является составной частью MS Office, и несмотря на высокую стоимость широко используется. Рабочее окно Visio 2013 представлено на рисунке 3.7.



**Рисунок 3.7 – Рабочая область Visio 2013**

 **Рисунок 3.8 – Эскиз главного меню**



**Рисунок 3.9 – Эскиз режима Создание центра**



**Рисунок 3.10 – Эскиз режима Создание объекта**



**Рисунок 3.11 – Эскиз режима Параметры мониторинга**



Рисунок 3.12 – Эскиз режима Шаг таймера



**Рисунок 3.13 – Эскиз режима Показать модель**

**3.5 Библиотека программ**

**3.6 Применение библиотеки**

**3.7 Выводы**

В третьей главе были получены следующие результаты:

* сформулированы требования к архитектуре системы;
* обоснован выбор системного и прикладного ПО: Mac OS X, Xcode, Objective-C, Cocoa;
* средствами специализированной системы Visio 2013 разработан комплект эскизов интерфейсов, соответствующих архитектуре;
* средствами языка Ruby разработана библиотека программ, реализующих архитектуру, на ее основе построена система Monitoring;
* проведена апробация системы на примере моделирования процесса организации мониторинга тягача и цистерны с токсичными отходами.