**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

Кафедра информационных систем управления

**РУДКОВСКИЙ ПАВЕЛ СЕРГЕЕВИЧ**

**СИСТЕМА МОДЕЛИРОВАНИЯ МОНИТОРИНГА МОБИЛЬНЫХ СТРУКТУРИРОВАННЫХ ОБЪЕКТОВ**

Отчет по преддипломной практике

студента 5 курса 2 группы

|  |  |
| --- | --- |
| “Допустить к защите“  с предварительной оценкой \_\_\_\_  **Руководитель работы**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  “\_\_” \_\_\_\_\_\_\_\_\_\_\_ 2014 г | **Руководитель**  *Вальвачев Александр Николаевич*  доцент кафедры ИСУ, к.т.н. |

Минск 2014

# РЕФЕРАТ

Дипломная работа, 41 с., 35 рис., 8 источников.

СИСТЕМА МОДЕЛИРОВАНИЯ МОНИТОРИНГА МОБИЛЬНЫХ СТРУКТУРИРОВАННЫХ ОБЪЕКТОВ

Объект исследования – модели, алгоритмы, системы мониторинга мобильных техногенных объектов.

Цель работы – изучить недостатки существующих подходов к мониторингу мобильных объектов, сформулировать задачу, обеспечивающую устранение недостатков, разработать модели, алгоритмы и программное обеспечение для автоматизации решения задачи мониторинга.

Методы исследования – методы организации мониторинга, методы распознавания образов, методы принятия решений, методы объектно-ориентированного подхода для разработки программных систем.

Результат– модели, алгоритмы и программный инструментарий для автоматизации мониторинга мобильных объектов сложной структуры.

Область применения– прикладные задачи мониторинга движения железнодорожных составов, автоколонн, караваном морских и речных судов.

**СОДЕРЖАНИЕ**

|  |  |  |  |
| --- | --- | --- | --- |
| ВВЕДЕНИЕ | | | 1 |
| 1 |  | АНАЛИЗ ПРОБЛЕМЫ И ПОСТАНОВКА ЗАДАЧИ | 3 |
|  | 1.1 | Основные понятия и определения | 3 |
|  | 1.2 | Анализ проблемы мониторинга подвижных объектов | 4 |
|  | 1.3 | Постановка задачи | 5 |
|  | 1.4 | Декомпозиция задачи | 6 |
|  | 1.5 | Выводы | 6 |
| 2. |  | МОДЕЛИ И АЛГОРИТМЫ | 7 |
|  | 2.1 | Требования к моделям и алгоритмам | 7 |
|  | 2.2 | Модели | 7 |
|  | 2.2.2 | Модель среды | 8 |
|  | 2.2.3 | Модель сцены | 8 |
|  | 2.2.4 | Модель объекта наблюдения | 9 |
|  | 2.3 | Алгоритмы | 9 |
|  | 2.3.1 | Алгоритм оценки состояния объекта | 10 |
|  | 2.3.2 | Алгоритм синтеза управления для объекта | 10 |
|  | 2.4 | Выводы | 11 |
| 3 |  | ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ | 12 |
|  | 3.1 | Требования к архитектуре и программному инструментарию | 12 |
|  | 3.2 | Выбор программной платформы | 12 |
|  | 3.3 | Архитектура системы | 15 |
|  | 3.4 | Эскизы интерфейсов | 17 |
|  | 3.5 | Библиотека программ СТЕНД | 20 |
|  | 3.6 | Применение библиотеки программ для решения практической задачи | 21 |
|  | 3.7 | Выводы | 22 |
| ЗАКЛЮЧЕНИЕ | | | 23 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | | | 23 |

**ВВЕДЕНИЕ**

Проблема мониторинга обострилась в связи с глобальными изменениями общества и климата. Особую актуальность приобрел мониторинг мобильных объектов (караванов судов, автоколонн и т.п.), т.к. они часто перевозят потенциально опасные грузы, повышающие вероятность техногенных катастроф. Эта тематика недостаточно исследована по причине новизны и быстрых изменений среды, вызвавших отставание теории от практики.

Понятие “мониторинг” впервые было сформулировано Р.Манном на Стокгольмской конференции по окружающей среде в 1972 г. как “система повторных наблюдений одного и более элементов окружающей природной среды в пространстве и во времени с определенными целями в соответствии с заранее подготовленной программой”. Был разработан широкий спектр методов наблюдения за природно-территориальными объектами (в основном заповедниками) и обработки данных.

В классическом понимании “мониторинг – это комплексный процесс, включающий наблюдение, оценку и прогноз изменений состояния биосферы под влиянием естественных и антропогенных факторов” [1]. В зависимости от масштаба наблюдений мониторинг подразделяется на глобальный, региональный, локальный. Как правило, мониторинг включает три основных составляющих: наблюдения за объектами, оценки их состояния по заданным признакам и выработки адекватного управления.

В 1990-2012 гг. в результате быстрых политических, социальных и технологических изменений, вызванных процессами глобализации, изменения климата и информатизации, возникла необходимость в адаптации традиционных методов и технологий мониторинга к свойствам новой глобальной среды. К ним, в первую очередь, относятся: высокий уровень неопределенности, динамизм жизненного цикла, распределенность, мобильность и сложность структуры объектов наблюдения [4, 5]. В таких условиях традиционные методы мониторинга теряют эффективность, соответственно, возрастает вероятность возникновения экологических, социальных и других катастроф.

В настоящее время мониторинг стали трактовать более широко как “комплекс задач наблюдения за мобильными техногенными объектами и земной поверхностью с целью управления их состоянием” [2]. Для автоматизации мониторинга используются специализированные системы мониторинга разрабатывают. Их разработкой и эксплуатацией занимаются, как правило, крупные организации, способные решать дорогостоящие и достаточно сложные организационные, теоретические и технологические проблемы. В то же время существует большое количество фирм-перевозчиков, не применяющих системы мониторинга из-за их высокой стоимости.

В данной работе сделана попытка разработать теоретическую и технологическую базу для разработки недорогих систем мониторинга, рассчитанных на средние и малые компании.

В данной работе решаются традиционные задачи мониторинга, адаптированные к условиям глобальной среды и свойства инфраструктуры Интернет:

* построение моделей среды, сцены мониторинга и объекта наблюдения;
* разработка алгоритмов оценивания состояния удаленного объекта и выработки соответствующего управляющего решения;
* разработка программного обеспечения и технологии, обеспечивающей моделирование, оценку и синтез управляющего решения.

Для автоматизации разработанных моделей и алгоритмов использовался язык программирования Objective-C.

**ГЛАВА 1 АНАЛИЗ ПРОБЛЕМЫ И ПОСТАНОВКА ЗАДАЧИ**

**1.1 Основные понятия и определения**

Решение любой проблемы начинается с построения понятийного каркаса, который однозначно понимается всеми участниками решения. Приведем комплекс основных определений, необходимых для автоматизации мониторинга [1, 3, 4, 7].

*Определение 1*. Мониторинг – это “комплекс задач наблюдения за естественными и искусственными объектами с целью управления их состоянием”.

Приведем комплекс вспомогательных определений, необходимых для автоматизации мониторинга.

*Определение 2*. Структурированный мобильный объект (СМО) – это мобильные технически сложные объекты, структура которых включает разнородные элементы. К типичным СМО относятся: железнодорожные составы, автоколонны, танкеры с горюче-смазочными материалами, радиоактивными и другими токсичными отходами. Такого рода СМО являются потенциальной причиной возникновения крупномасштабных чрезвычайных ситуаций, поэтому мониторинг СМО является важнейшей задачей. К МО так же относится водный, автомобильный и гусеничный транспорт государственных и частных компаний, от эффективности деятельности которого зависит успех их экономической деятельности.

*Определение 3*. Жизненный цикл (ЖЦ) СМО – это процессы создания, использования и расформирования СМО, относящиеся к реализации определенного проекта.

*Определение 4.* Среда– совокупность природных и искусственных условий, в которых реализуется жизненный цикл СМО. Среды, в которых функционирует СМО, могут отличаться специфическими климатическими, ландшафтными, коммуникационными и другими свойствами.

*Определение 5.* Сцена – группа одушевленных и искусственных участников (акторов) мониторинга, использующая каналы связи для обмена информацией в процессе решения общей задачи. *Определение 6.* Траектория – совокупность точек в средах, по которым движется СМО.

*Определение 6*. Контрольные точки – точки на траектории, в которых измеряются значения диагностических показателей.

*Определение 7*. Диагностические показатели – множество переменных, значения которых характеризуют различные свойства СМО и могут фиксироваться контроллером.

*Определение 8*. Регистратор – устройство, обеспечивающее фиксацию и передачу по каналам связи значений диагностических показателей.

*Определение 9*. Состояние объекта – строковая константа, зависящая от значений диагностических показателей и характеризующая возможность объекта реализовать проект в данный момент времени.

*Определение 10*. Управляющее решение – строковая константа, соответствующая состоянию и содержащая рекомендации (указания) для изменения состояния объекта в данный момент времени.

На основе введенных определений и аксиом можно формально описать задачу мониторинга СМО.

**1.2 Анализ проблемы мониторинга подвижных объектов**

Как показано выше, основная проблема СМО заключается в том, что они являются причиной техногенных катастроф. Устранение границ привело к резкому увеличению количества автоколонн и железнодорожных составов, сухогрузов и танкеров, маршруты которых проходят в малоисследованных местах и быстроизменяющейся географической, политической и социальной обстановке [1].

ЛПР СМО в быстро изменяющейся обстановке не всегда успевает правильно реагировать, поэтому возникают критические ситуации.

Для предотвращения таких ситуаций и быстрой реакции при их возникновении, необходимо, разработать методы и технологии для постоянного мониторинга потенциально опасных движущихся технических систем. Потенциальная возможность решения этой задачи появилась после развития глобальной сети Интернет и недорогих высокоэффективных средств беспроводной телефонной связи и глобального позиционирования.

Решение проблемы автоматизации мониторинга МСО включает решение следующих проблем [1]:

* построение моделей сцены мониторинга, центра и объекта наблюдения;
* разработка алгоритмов оценивания состояния удаленного объекта и выработки соответствующего управляющего решения;
* разработка программного обеспечения и технологии, обеспечивающей моделирование, оценку и синтез управляющего решения для объекта наблюдения.

Для комплексного решения поставленных проблем необходимо сформулировать общую задачу дипломной работы.

**1.3 Постановка задачи**

При постановке задачи необходимо учитывалась возможность ее отображения на множество реальных ситуаций, поэтому как постановку, так и предлагаемое решение можно считать в некотором смысле типовыми для задач мониторинга. При постанове задачи используем онтологический подход, т.к. краткое описание составляющих решение в рамках некотрой иерархии [5].

Дано: организация W, в состав которой входят центр управления H и n мобильных объектов управления G = G1,G2,…,Gn. Объекты (например, тепловоз и n железнодорожных цистерн) участвуют в решении некоторой задачи, (например, в перевозке горюче-смазочных материалов). Успех решения задачи зависит от состояния объектов.

Состояние объектов G характеризуется конечным множеством диагностических переменных:

X = (X1,X2,…,Xm).

Объекты G территориально распределены и меняют свое географическое положение KtG в зависимости от времени:

Kt1G ≠Kt2G.

Для оценки текущего состояния объектов руководство организации W должно периодически решать задачу мониторинга M, включающую оценку состояния (V) объектов (G) на основе текущих значений переменных X и синтез соответствующего управляющего решения (U):

M🡪U = f (X,V).

Требуется разработать архитектуру системы Sys, обеспечивающую автоматизацию решения задачи мониторинга M.

Требование к решению: инвариантность времени мониторинга к количеству объектов (t🡪 0 при n🡪∞).

В основу решения положим следующие соображения:

* особенность поставленной задачи заключается в географической распределенности объектов наблюдения и, соответственно, распределенности информации об их состоянии. Для такого рода задач в большинстве случаев используется распределенный подход [7jy];
* в соответствии с процессным подходом решение поставленной задачи можно свести к пяти процессам: построение организации W, получение информации X от объектов, передачу X в центр, интеграцию X в БД центра, обработку X для оценки V объекта и выработки соответствующего управления U.

**1.4 Декомпозиция задачи**

Решение общей задачи предлагается разделить на четыре подзадачи:

* построение модели сцены и участников;
* разработка алгоритмов мониторинга;
* разработка программного обеспечения;
* разработка методики применения программного обеспечения и решение на ее основе практической задачи.

**1.5 Выводы**

В первой главе получены следующие результаты:

* даны определения для основных понятий, используемых в процессе мониторинга структурированных мобильных объектов (СМО);
* выявлены проблемы, связанные с автоматизацией мониторинга;
* сформулирована задача на разработку программного обеспечения для поддержки мониторинга СМО;
* выполнена декомпозиция общей задачи на подзадачи, обеспечивающее в комплексе получение искомого решения.

**ГЛАВА 2. МОДЕЛИ И АЛГОРИТМЫ**

**2.1 Требования к моделям и алгоритмам**

Основное требование к моделям заключается в возможности их представления в форме, допускающей их последовательное усовершенствования от уровня концепции до уровня программного кода на любом объектно-ориентированном языке. Кроме того, модель должна быть наглядной (допускать визуализацию) и интуитивно понятной как руководству (центру), так и ЛПР объектов наблюдения.

Основными требованиями к алгоритмам являются:

* использование в качестве основы известной и хорошо апробированной теоретической базы;
* возможность реализации с помощью средств современных программных платформ.

Ниже представлены модели и алгоритмы, соответствующие этим требованиям.

**2.2 Модели**

Для создания теоретического базиса решения предлагается комплекс моделей, построенных на основе онтологического подхода, который формализован в стандарте IDEF5 и активно применяется при решении сложных задач автоматизации на кафедре ИСУ БГУ.

Онтология – это формальная спецификация разделяемой концептуальной модели [5]. Онтология состоит из классов сущностей предметной области, свойств этих классов связей между этими классами и утверждений, построенных из этих классов, их свойств и связей между ними. Далее онтологические модели для краткости будем называть моделями.

Согласно логике задачи мониторинга, прежде всего, необходимо построить модель сцены мониторинга, в рамках которой он будет проводится.

Модель сцены, должна, как минимум, в качестве составляющих должна включать атрибуты центра (H) и объектов (G):

W = (H, G1, G2, …,Gn) (2.1)

Модель центра включает атрибуты глобальной идентификации, список задач мониторинга, сервер, программу обработки F показателей X и базу данных для хранения данных об объекте наблюдения:

H = (AH, idH, task, S, PL, DB), (2.2)

где: AH – глобальный адрес центра; idH – идентификатор центра; task – задачи мониторинга; S – сервер; PL – программы синтеза состояний и управлений; DB – база данных для аккумуляции показаний датчиков.

За методологическую основу задачи мониторинга возьмем хорошо исследованную задачу принятия решений []:

U = F (X,V), (2.3)

где: F – функция выбора управления U для объекта в зависимости от его значений диагностических переменных X и состояния V.

Модель объекта наблюдения включает составляющие, необходимые для идентификации объекта в сети Интернет, сбора данных об объектах и их передачи в центр для обработки:

G = (AG, idG, KG, d1,d2,…, dm, Ks, Box, Kp), (2.4)

где: AH – глобальный адрес объекта; idG – идентификатор объекта; G – текущие координаты; d – датчики; Ks – контроллер для снятия и анализа сигналов датчиков; Box – сообщение от объекта к центру; Kp – контроллер для обмена данными между объектом и центром.

Модель пакета (сообщения) включает адреса получателя, отправителя, идентификационные атрибуты объекта и показания, снятые с датчиков:

Box = (AH,AG, idG, X) (2.5)

В совокупности модели 1-6 представляют собой основу для решения задачи. Использование онтологического подхода позволяют уточнять их до уровня программного кода. Общая схема реализации мониторинга представлена на рисунке 2.1.

**2.3 Алгоритмы**

2.3.1 Алгоритм построения сцены мониторинга

Алгоритм построение сцены формируется на основе моделей центра (2.2) и объекта наблюдения (2.4).

На входе алгоритма:

– AH, idH, task, S, PL, DB – реквизиты центра и пустая база данных,

– AG, idG, KG, d1,d2,…, dm, Ks, Box, Kp – реквизиты объекта наблюдения.

На выходе алгоритма:

– <DB> - заполненная база данных

– X,V,U – реквизиты проекта (задачи мониторинга)



**Рисунок 2.1 – Схема алгоритма построения сцены мониторинга СМО**

Данный алгоритм формирует базу данных, соответствующую сцене мониторинга, что дает основание для разработки алгоритма синтеза управляющего решения.

2.3.2 Алгоритм оценки и синтеза управляющего решения для объекта

наблюдения

Алгоритм синтеза оценки строится на основе базы данных <DB> и текущего вектора данных <X>, характеризующего объект наблюдения в контрольных точках.

На входе алгоритма:

– X,V,U,<X> – реквизиты проекта и вектор значений диагностических переменных.

На выходе алгоритма:

– <V>,<U> – состояние и управляющее решение для объекта наблюдения.



**Рисунок 2.2 – Схема алгоритма мониторинга СМО**

Данный алгоритм формирует оценку и решение, как и требовалось для теоретического решения задачи, что позволяет перейти к стадии разработки соответствующего программного обеспечения.

**2.4 Выводы**

Во второй главе получены следующие результаты:

* разработаны модели участников сцены мониторинга, включая модели центра, объекта наблюдения, сообщения и средств для обмена информацией;
* разработаны алгоритмы оценки состояния объекта наблюдения и синтеза соответствующего управляющего решения.

**ГЛАВА 3 ПРОГРАММОЕ ОБЕСПЕЧЕНИЕ**

**3.1 Этапы разработки программного обеспечения**

От правильности выбора системного и прикладного программного обеспечения для разработки систем мониторинга зависит время, трудоемкость, эффективность и стоимость целевой системы.

Согласно современным требованиям [5,8] построение компьютерных систем включает, как минимум, четыре основных этапа:

* выбор операционной системы, согласно требованиям проекта;
* выбор программной платформы и языка, соответствующего целям задачи;
* разработка архитектуры ПО (компонент и связей между ними);
* реализация архитектуры (выбор программной платформы, программная реализация компонент, тестирование, описание этапов применения технологии);
* внедрение (использование ПО для решения прикладных задач).

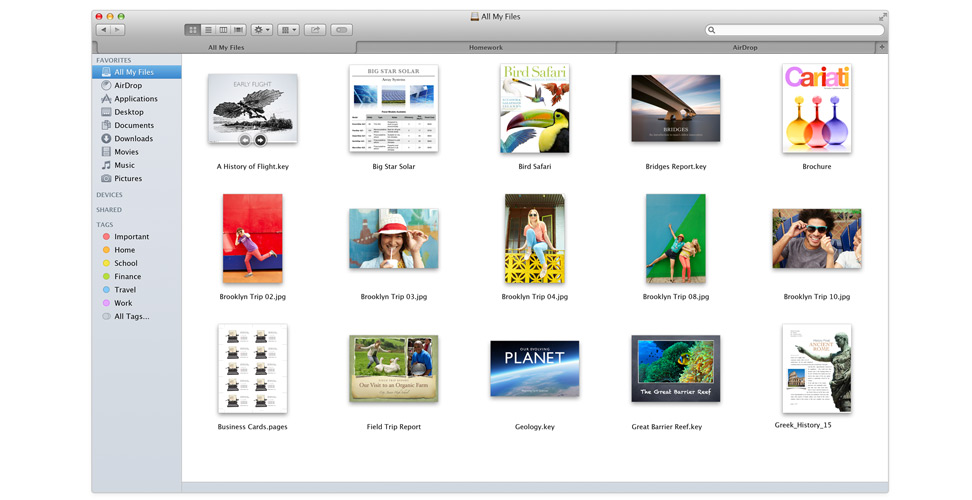
**3.2 Выбор программной платформы**

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.

В данной работе в качестве операционной системы была выбрана Mac OS X, которая значительно отличается от предыдущих версий Mac OS. Для реализации программной части использовался язык Objective-C в связке с фрэймворком Cocoa. Разработка велась в среде Xcode 5.1.1 [8]

**3.3 Обзор Mac OS X**

Основа системы — POSIX-совместимая операционная система Darwin, являющаяся свободным программным обеспечением. Её ядром является XNU (рекурсивный акроним от «X is Not Unix» — «X не Unix»), в котором используется ядро Mach и стандартные сервисы BSD. Все возможности Unix в OS X доступны через консоль.



**Рисунок 3.1 – Окно OS X**

Поверх этой основы в Apple разработано много проприетарных компонентов, таких как API Cocoa и Carbon, Quartz.

В OS X используется (как впрочем в любой UNIX-системе) вытесняющая многозадачность и защита памяти, позволяющие запускать несколько процессов, которые не могут прервать или повредить друг друга. На архитектуру OS X повлияла OpenStep, которая была задумана как портируемая операционная система. К примеру, NeXTSTEP была портирована с оригинальной платформы 68k компьютера NeXT, до того как NeXTSTEP была куплена Apple. Так и OpenStep была портирована на PowerPC в рамках проекта Rhapsody.

Основами OS X являются:

* подсистема с открытым кодом — Darwin (ядро Mach и набор утилит BSD).
* среда программирования Core Foundation (Carbon API, Cocoa API и Java API).
* графическая среда Aqua (QuickTime, Quartz Extreme и OpenGL).
* технологии CoreImage, CoreAudio и CoreData.

OS X также включает среду разработки программного обеспечения Xcode, которая позволяет разрабатывать программы на нескольких языках, включая Си, C++, Objective-C, Ruby, Java, Object Pascal. Она поддерживает компиляцию в так называемые «универсальные программы» (Universal Binary), которые могут запускаться на нескольких платформах (x86, PowerPC), так же, как «fat binaries» использовались для запуска одного приложения на 68k и PowerPC платформах.

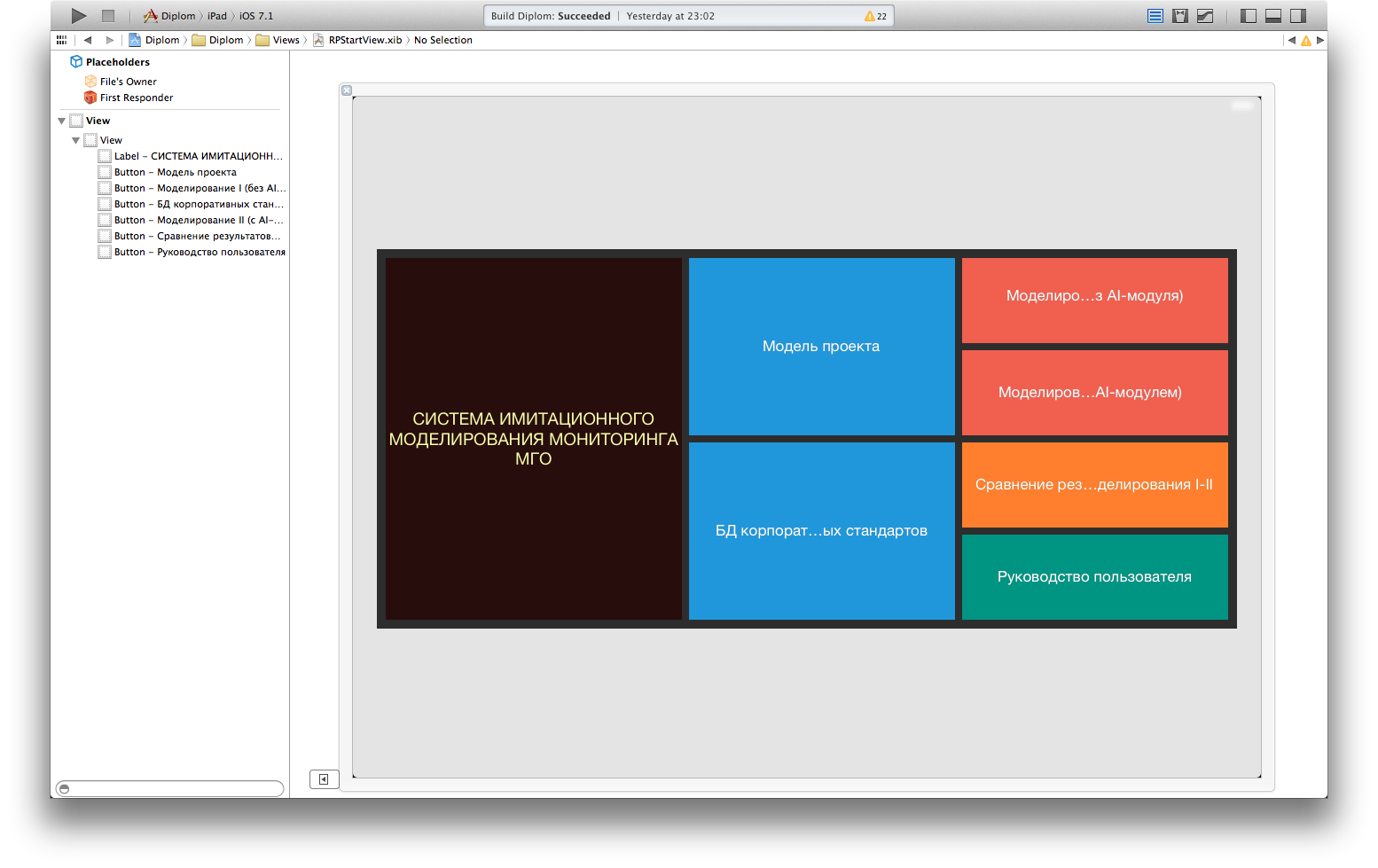
**3.4 Обзор среды разработки Xcode**

Xcode — программа для разработки приложений под OS X и iOS, разработанная компанией Apple. Поставляется бесплатно на установочном диске OS X вместе с операционной системой, но устанавливается пользователем вручную. Последняя версия — 5.1.1, не поддерживается старыми версиями OS X. Последнюю версию Xcode можно бесплатно загрузить из Mac App Store.

Основным приложением пакета является встроенная среда разработки, которая называется Xcode. Помимо этого, пакет Xcode включает в себя большую часть документации разработчика от Apple и Interface Builder — приложение, использующееся для создания графических интерфейсов.

Пакет Xcode включает в себя изменённую версию свободного набора компиляторов GNU Compiler Collection (GCC, apple-darwin9-gcc-4.0.1, который с июля 2012 года больше не будет существовать в составе инструментария разработчиков для OS X (в Xcode 4.4), осенью GCC покинет и iOS (его не будет в Xcode 4.5) и поддерживает языки C, C++, Objective-C, Objective-C++, Java, AppleScript, Python и Ruby с различными моделями программирования, включая (но не ограничиваясь) Cocoa, Carbon и Java.

Interface Builder упрощает создание пользовательского интерфейса (UI) (рисунок 3.2). С его помощью можно легко, без написания кода, создать слои из окон, различные кнопки, ползунки и другие элементы управления. Затем надо превратить этот прототип UI в реальное приложение, добавив новые возможности. Xcode работает с Interface Builder в режиме реального времени, так что в графическом интерфейсе (Interface Builder) видно то, что программист пишет в Xcode.



**Рисунок 3.2 – Интерфейс Builder**

Можно легко создавать пользовательские интерфейсы, поскольку Cocoa был построен с использованием шаблона Model-View-Controller (MVC). На самом деле, UI фактически являются архивами объектов Cocoa, которые не требуют генерации кода. Изменения в интерфейсе пользователя (UI) не требуют перекомпиляции (перепроверки) кода, а изменения в коде, не требуют перекомпиляции UI.

**3.5 Обзор языка Objective-C**

Objective-C, известный также как Objective C, ObjC или Obj-C — компилируемый объектно-ориентированный язык программирования, используемый корпорацией Apple, построенный на основе языка Си и парадигм Smalltalk. В частности, объектная модель построена в стиле Smalltalk — то есть объектам посылаются сообщения.

Язык Objective-C является надмножеством языка Си, поэтому Си-код полностью понятен компилятору Objective-C.

Компилятор Objective-C входит в GCC и доступен на большинстве основных платформ. Язык используется в первую очередь для Mac OS X (Cocoa) и GNUstep — реализаций объектно-ориентированного интерфейса OpenStep. Также язык используется для iOS (Cocoa Touch).

ObjC был создан Брэдом Коксом в начале 1980-х в его компании Stepstone. Он пытался решить проблему повторного использования кода.

Целью Кокса было создание языка, поддерживающего концепцию software IC. Под этой концепцией понимается возможность собирать программы из готовых компонентов (объектов), подобно тому как сложные электронные устройства могут быть легко собраны из набора готовых интегральных микросхем.

При этом такой язык должен быть достаточно простым и основанным на языке С, чтобы облегчить переход разработчиков на него.

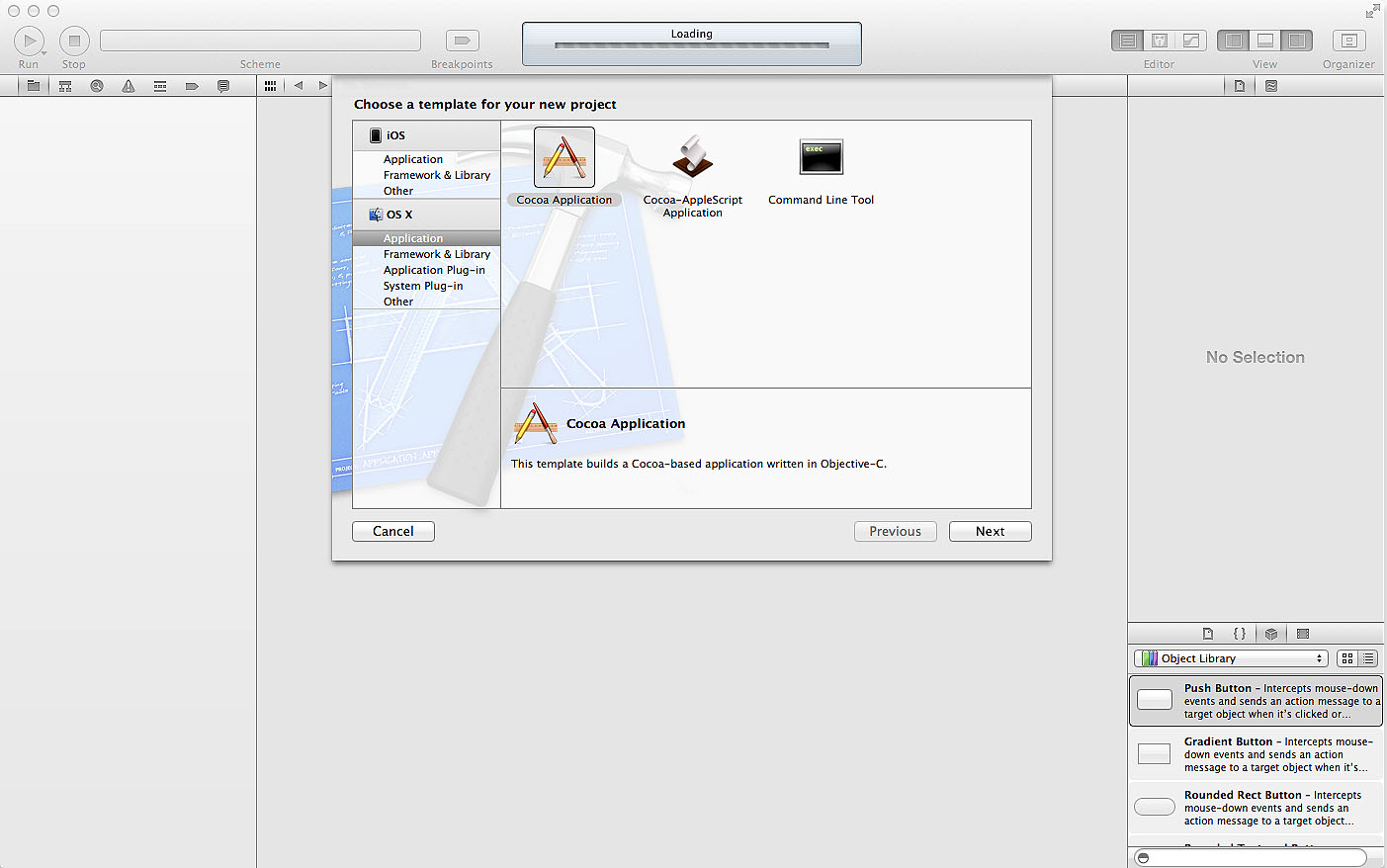
Одной из целей было также создание модели, в которой сами классы также являются полноценными объектами, поддерживалась бы интроспекция и динамическая обработка сообщений.

Получившийся в результате язык Objective-C оказался крайне прост — его освоение у С-программиста займет всего несколько дней. Он является именно расширением языка С — в язык С просто добавлены новые возможности для объектно-ориентированного программирования. При этом любая программа на С является программой и на Objective-C.

Одной из отличительных черт Objective-C является его динамичность — целый ряд решений, обычно принимаемых на этапе компиляции, здесь откладывается непосредственно до этапа выполнения.

**3.6 Обзор фрэймворка Сocoa**

Cocoa — родная объектно-ориентированная среда разработки приложений для операционной системы Mac OS X производства компании Apple (рисунок 3.3). Это один из пяти основных API, доступных в Mac OS X, — Cocoa, Carbon, Toolbox (для работы старых приложений Mac OS 9), POSIX и Java. Такие языки, как Perl, Python и Ruby не считаются основными, так как на них пока что пишется не так много серьёзных приложений для Mac OS X.



**Рисунок 3.3** – **Окно Cocoa**

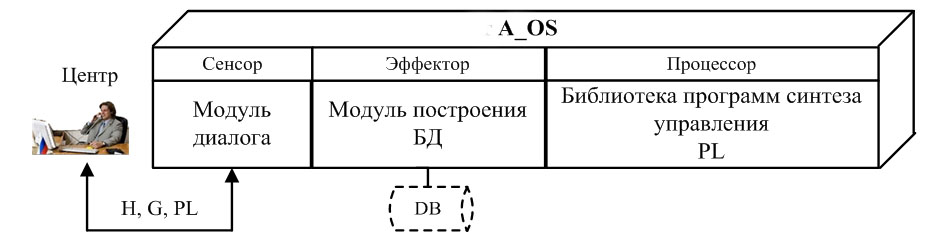
Приложения, использующие Cocoa, обычно разрабатываются с помощью среды разработки Apple Xcode (в прошлом называвшегося Project Builder) и Interface Builder с использованием языка Objective-C. Однако, среда Cocoa также доступна и при разработке на других языках, таких как Ruby, Python и Perl с помощью связующих библиотек (MacRuby, PyObjC и CamelBones соответственно). Также можно писать Cocoa-программы на Objective-C в обычном текстовом редакторе и вручную компилировать их с помощью GCC или make-сценариев для GNUstep.

С точки зрения конечного пользователя, Cocoa-приложения это приложения, написанные с использованием программной среды Cocoa. Такие приложения обычно имеют характерный внешний вид, поскольку эта среда во многом упрощает поддержку принципов «человечного интерфейса» Apple (Apple Human Interface Guidelines).

**3.7 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.

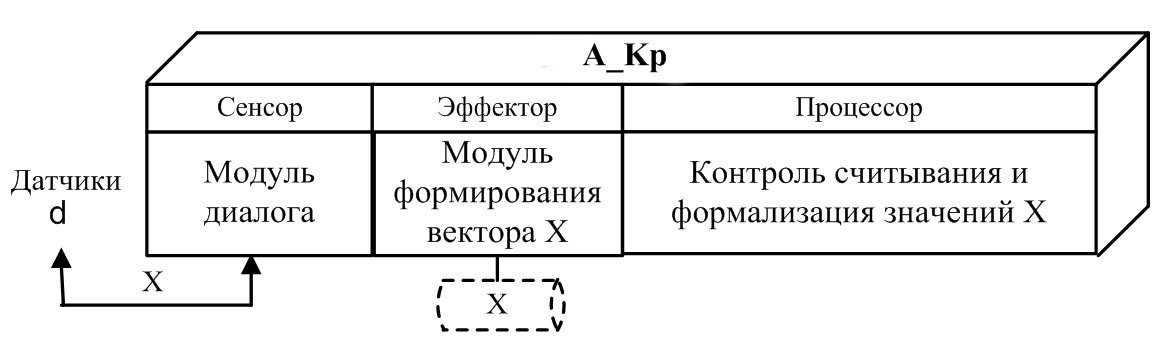
Синхронизация процессов осуществляется руководством организации, поэтому в первую очередь построим архитектуру модуля, формирующего модель иерархии и включающий атрибуты всех одушевленных и виртуальных (программы, устройства) участников организации(Рисунок 3.4).

****

**Рисунок 3.4** – **Архитектура модуля построения сцены**

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.

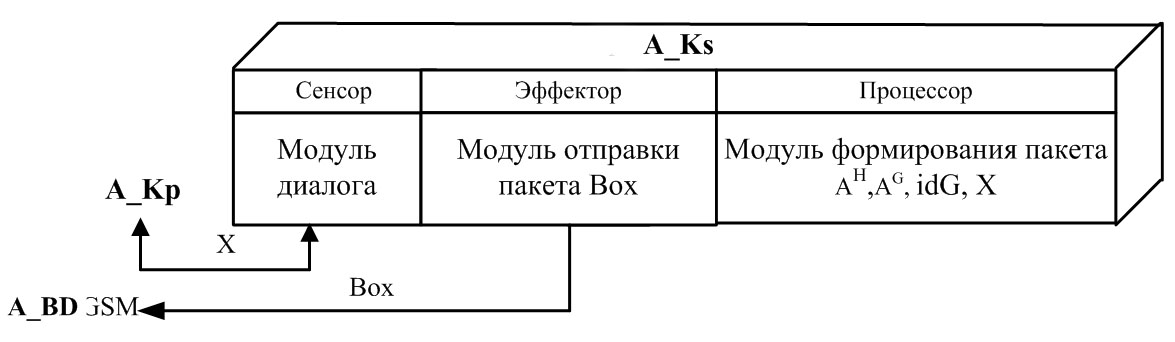
Первичным источником информации в мониторинге данного типа являются датчики. Соответственно, второй модуль должен получить значения признаков, формализовать их и передать агенту связи. Соответствующая архитектура представлена на рис.3.5.



**Рисунок 3.5** – **Архитектура модуля для формирования вектора значений диагностических 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.

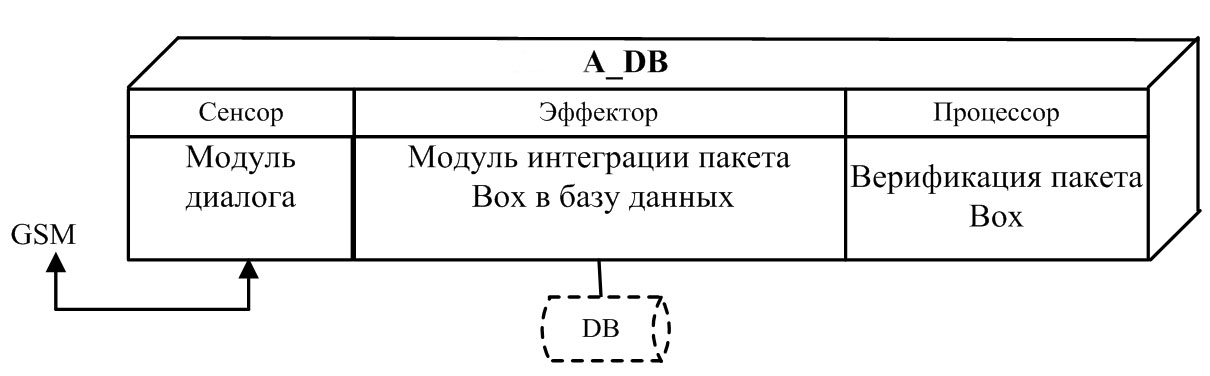
Далее активизируется модуль, который получает вектор X и формирует пакет-сообщение Box о текущем состоянии объекта для отправки в центр. Пакет содержит вектор показателей и идентификационные атрибуты объекта. Архитектура модуля A\_Ks, выполняющего эти операции, представлена на рис. 3.6.

** Рисунок 3.6** – **Архитектура модуля для формирования сообщения в центр**

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 3.7.

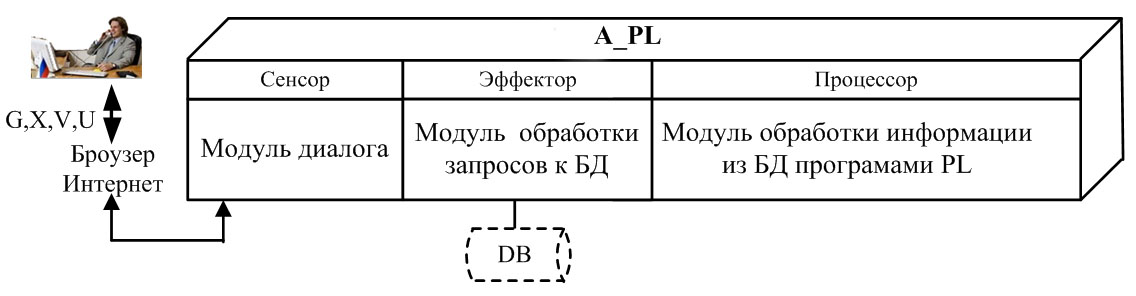
****

**Рисунок 3.7** – **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. 3.8.

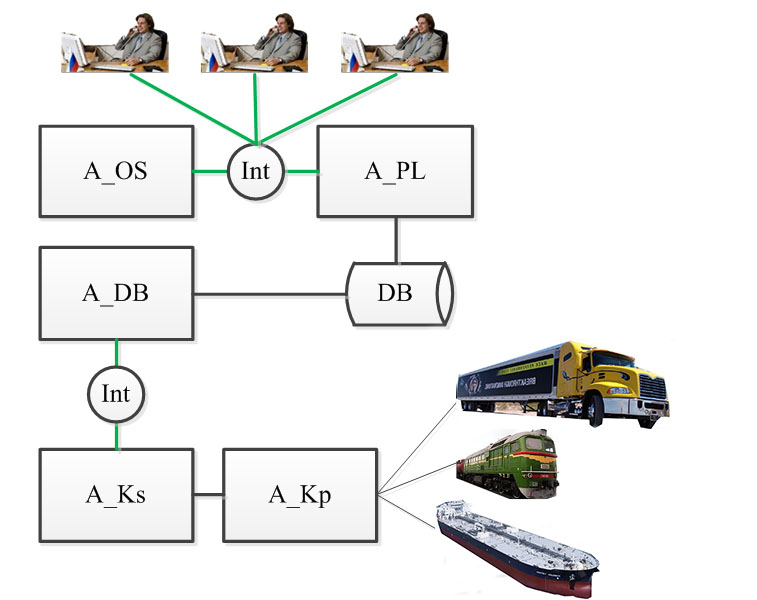
****

**Рисунок 3.8** – **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.3.9.



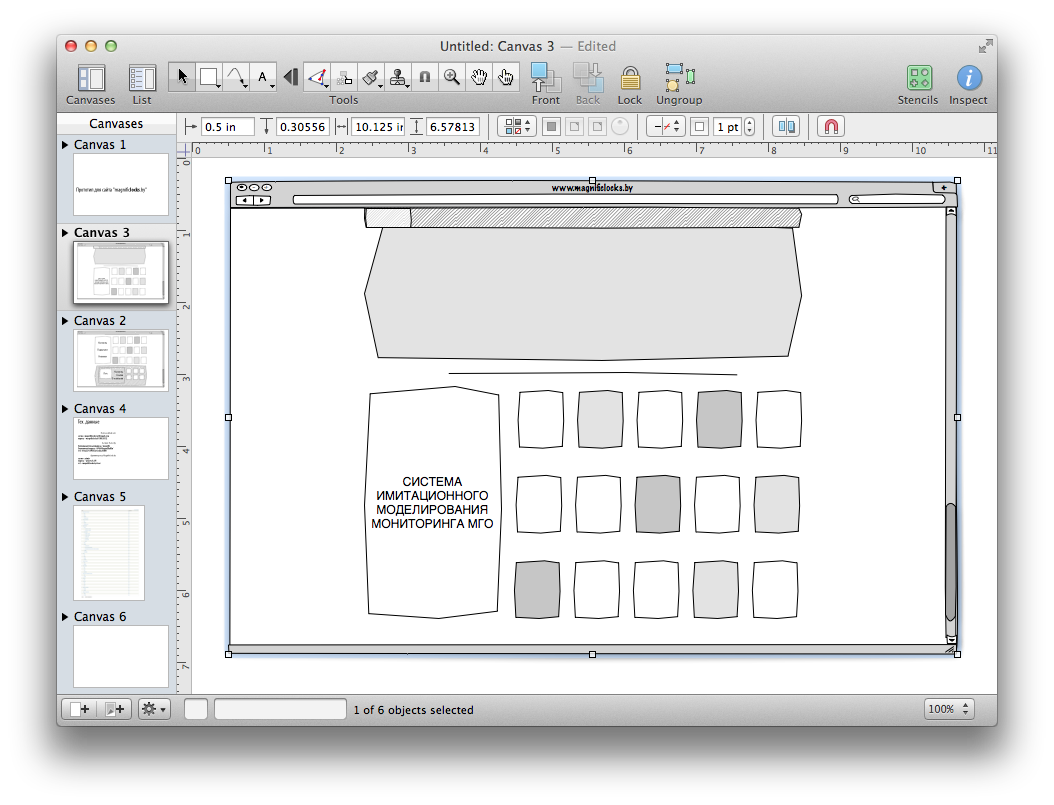
**Рисунок 3.9** – **Типовая архитектура системы мониторинга СМО**

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.

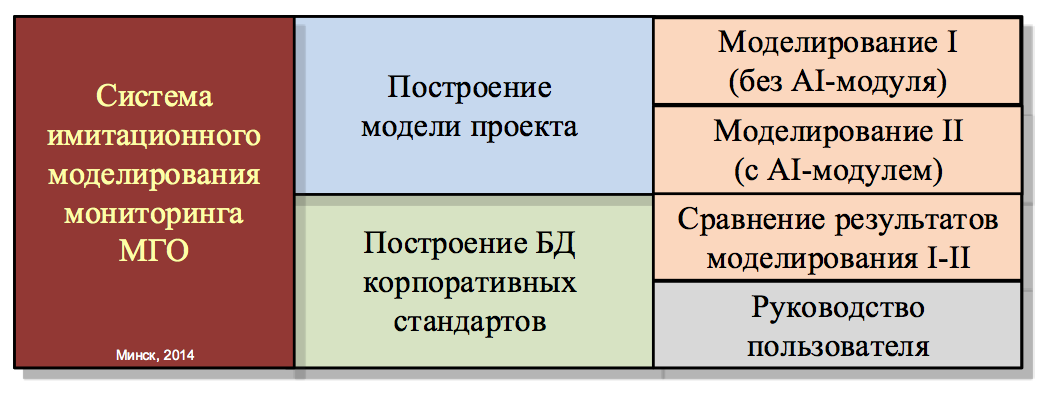
**3.8 Эскизы интерфейсов**

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 OmniGraffle, которая, несмотря на высокую стоимость широко используется. Рабочее окно OmniGraffle представлено на рисунке 3.10.



**Рисунок 3.10 – Рабочая область OmniGraffle**

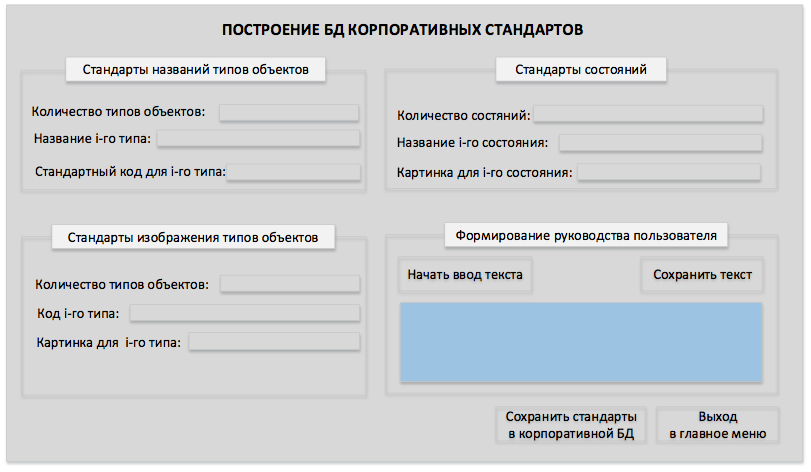
Разработанные эскизы реализации алгоритмов и визуализации результатов представлены на рисунках 3.11- 3.17.



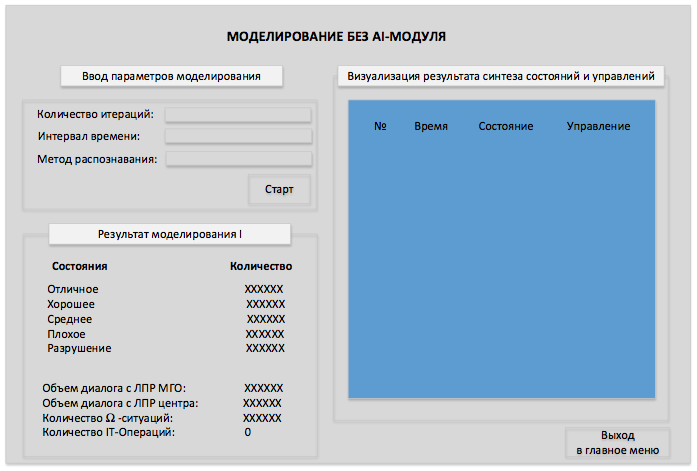
**Рисунок 3.11 – Эскиз главного меню**



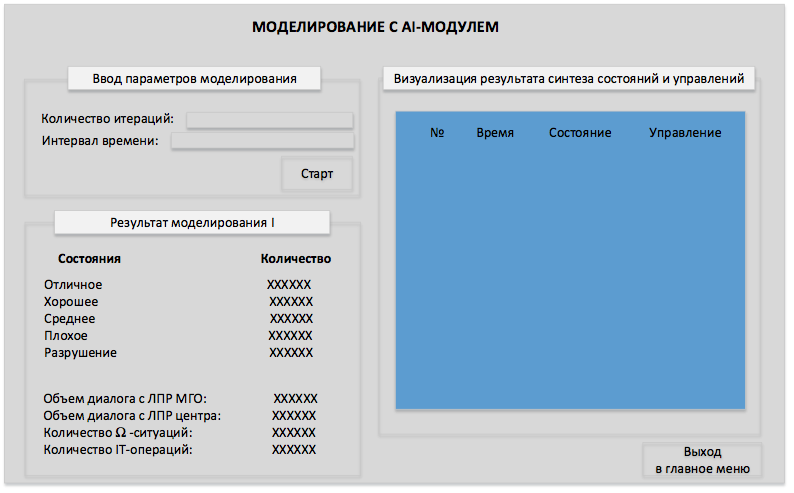
**Рисунок 3.12 – Эскиз режима построения модели**



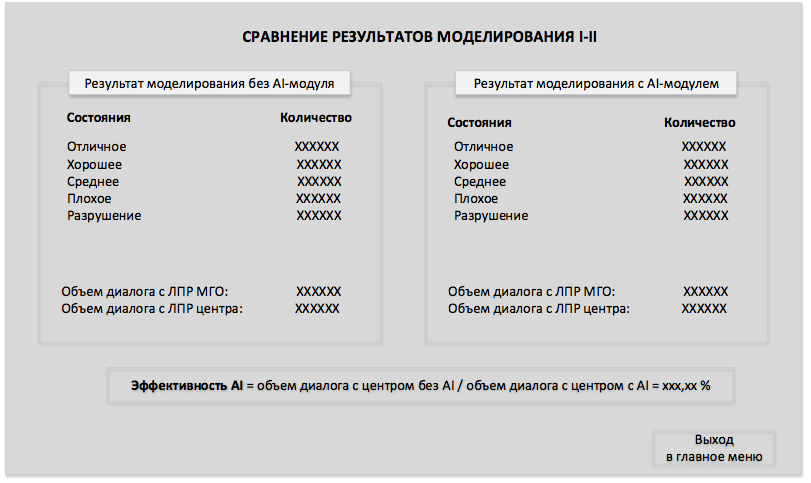
**Рисунок 3.13 – Эскиз режима создания стандартов**



**Рисунок 3.14 – Эскиз режима моделирования без AI-модуля**



**Рисунок 3.15 – Эскиз режима моделирования c AI-модулем**



**Рисунок 3.16 – Эскиз режима сравнения результатов**

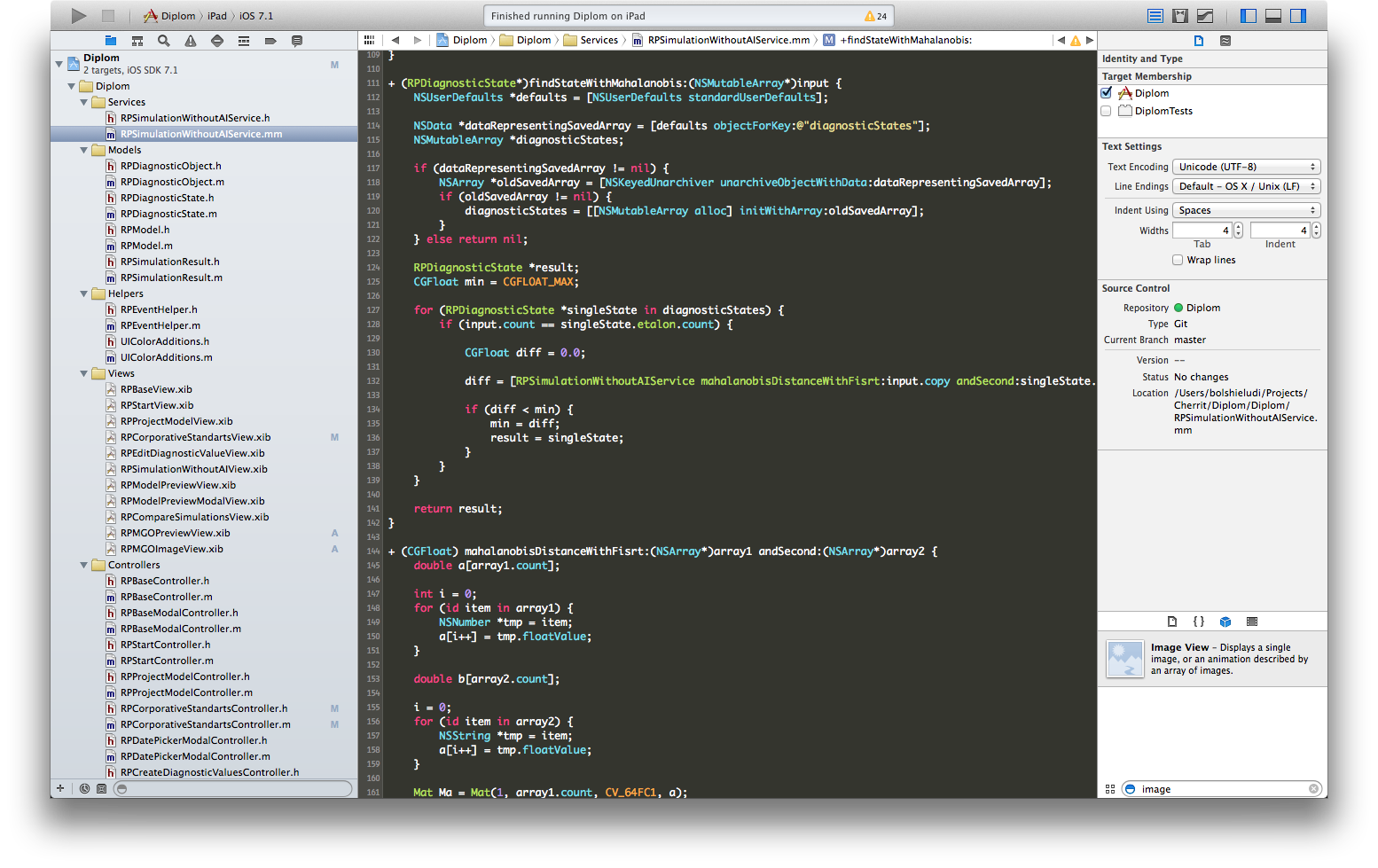


**Рисунок 3.17 – Эскиз руководства пользователя**

Приведенные выше эскизы могут быть реализованы в рамках различных подходов, например, в стиле Metro, развитом компанией Microsoft в Windows 8, или в любом другом стиле. В данной работе предлагается минималистический стиль мобильной операционной системы iOS 7, что обеспечивает наглядность выполнения всех этапов мониторинга в рамках 1-2 окон.

**3.9 Библиотека программ**

В процессе реализации архитектуры с использованием Mac OS X, Xcode, Objective-C, Cocoa (рисунок 3.18) была разработана библиотека Monitoring.



**Рисунок 3.18 – Окно разработки программ на языке Objective-C**

Библиотека содержит пять программных модулей:

- модуль построения сцены мониторинга;

- модуль анализа контрольных точек и формирования вектора значений диагностических показателей для объекта;

- модуль оценки объекта и синтеза соответствующего управляющего решения;

- модуль отправки сообщения в центр.

Библиотека реализована на языке Objective-C для Apple iPad. Она так же может быть реализована на любом современном объектно-ориентированном языке.

На основе библиотеки был разработан вариант системы для моделирования мониторинга СМО, главное меню которого представлено на рисунке 3.19.

****

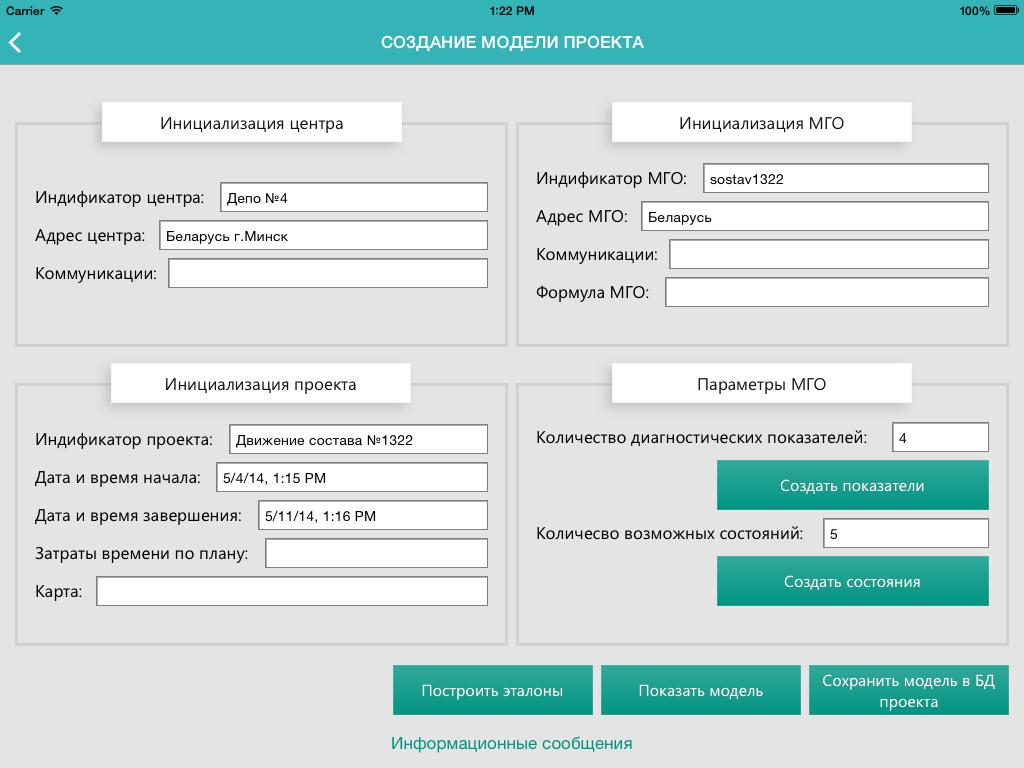
**Рисунок 3.19 – Главное меню системы**

Методика применения разработанной системы описана ниже.

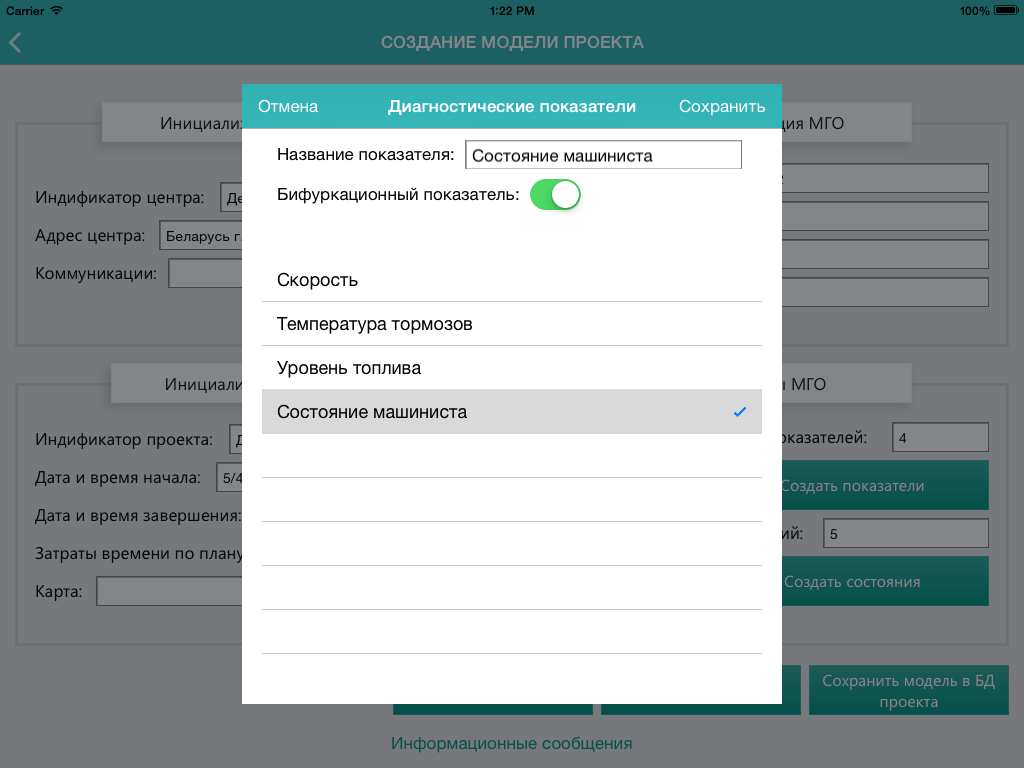
**3.10 Методика применения системы**

Методика применения системы включает пять шагов.

**Шаг 1.** Построение сцены (рисунок 3.20), задание диагностических показателей (рисунок 3.21) и диапазона их изменения. В сцене участвуют ЛПР Железнодорожного Депо 4 и железнодорожный состав из тепловоза и цистерны.

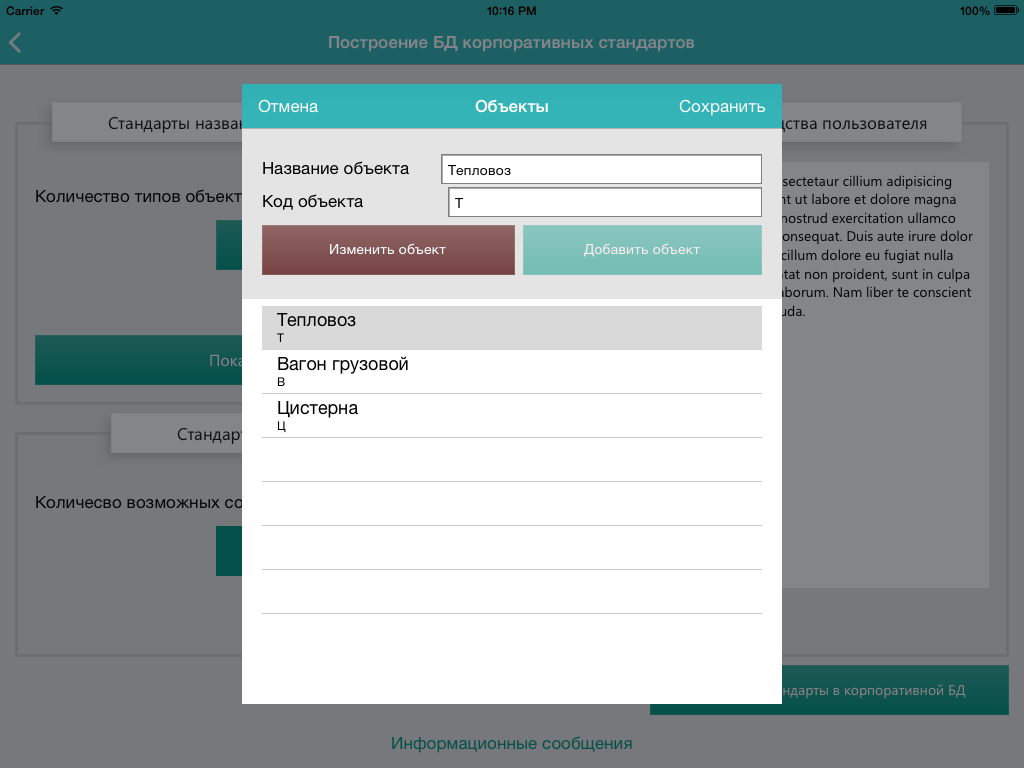


**Рисунок 3.20 – Построение сцены.**

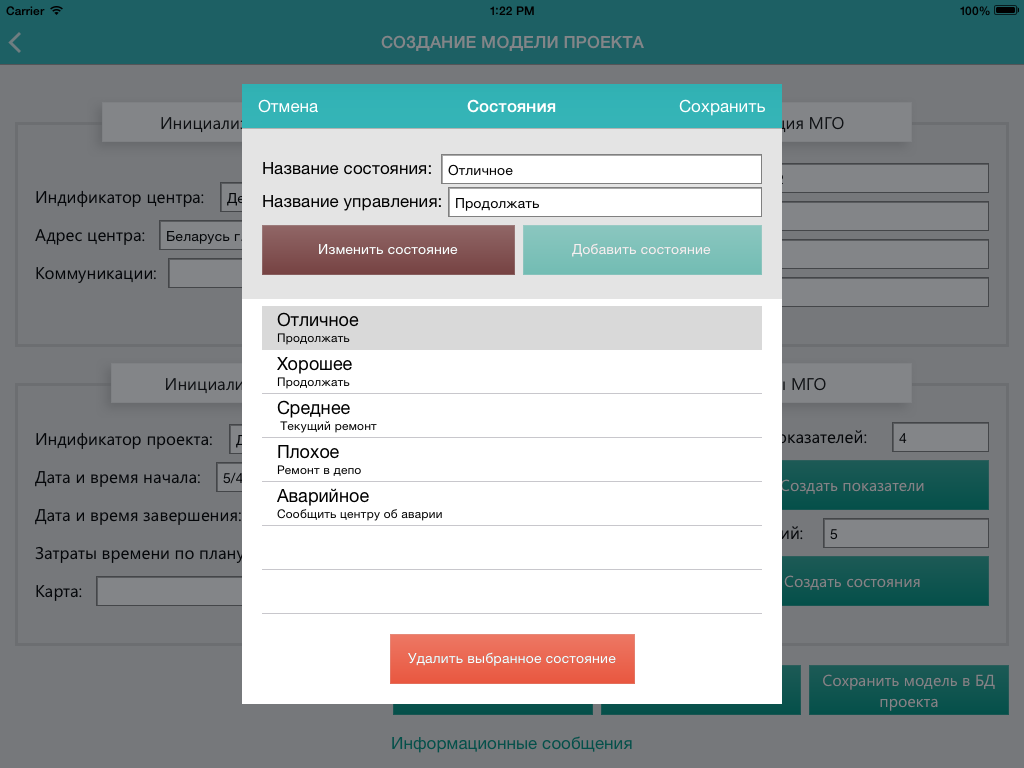


**Рисунок 3.21 – Задание диагностических показателей**

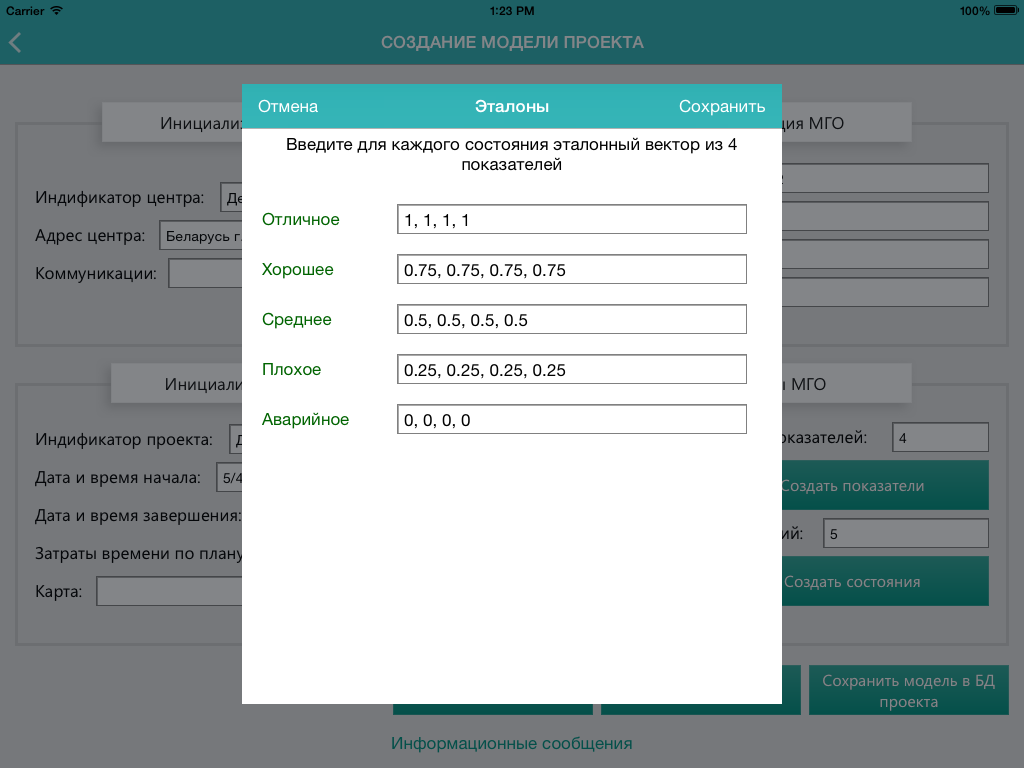
**Шаг 2**. Построение объектов (рисунок 3.22), состояний (рисунок 3.23), эталонов (рисунок 3.24), управлений.



**Рисунок 3.22 – Создание объектов**

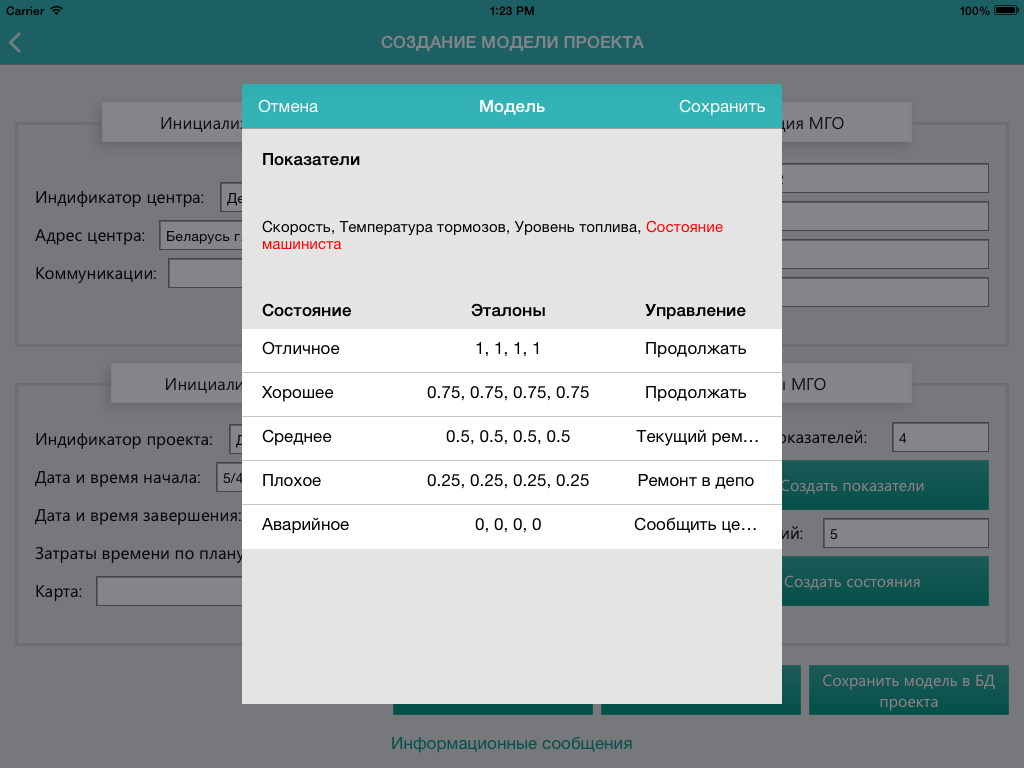


**Рисунок 3.23 – Ввод состояний**

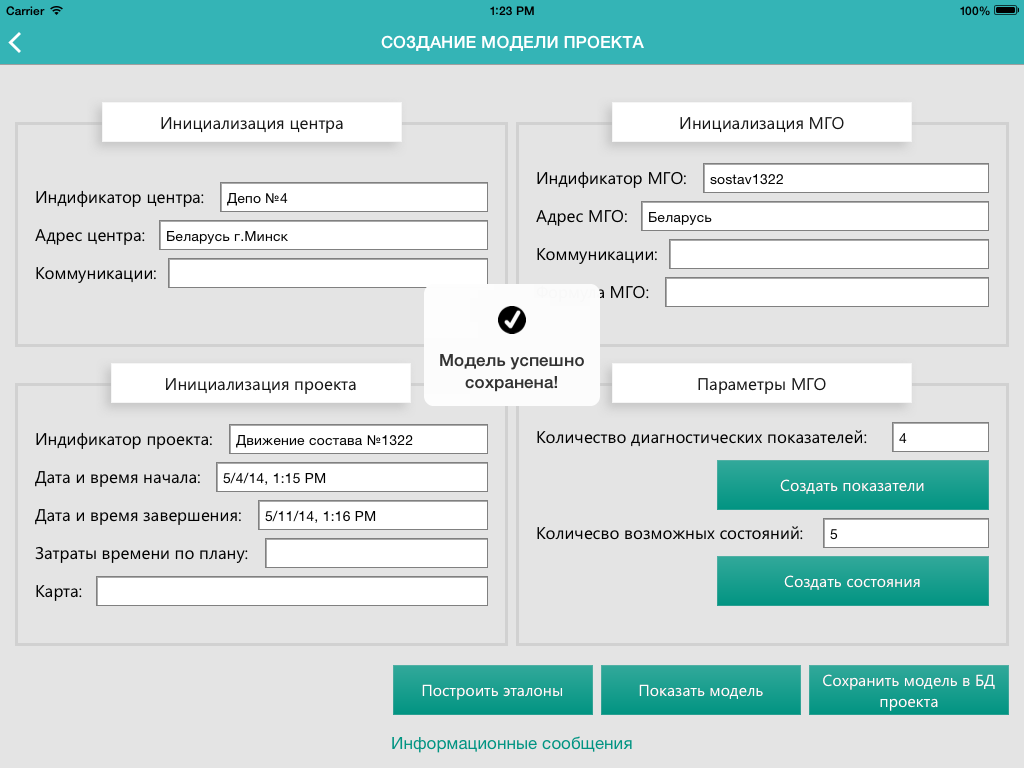


**Рисунок 3.24 – Ввод состояний**

**Шаг** **3**. Создание (рисунок 3.25) и сохранение модели в БД приложения (рисунок 3.26).

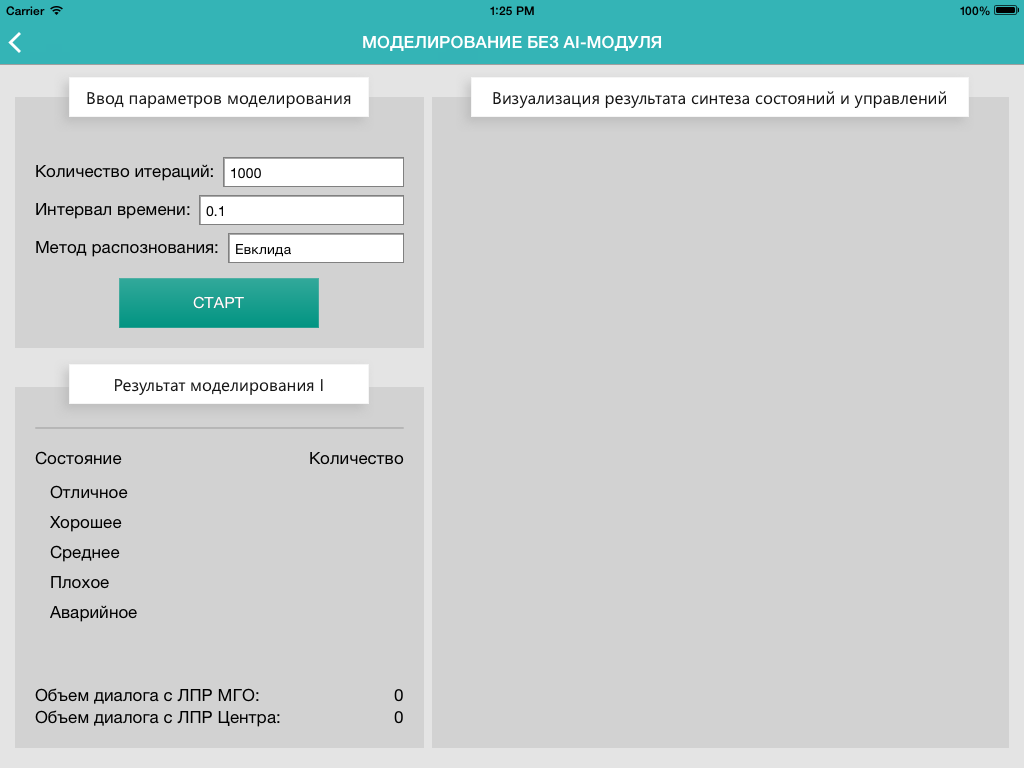


**Рисунок 3.25 – Созданные состояния, эталоны и управления**

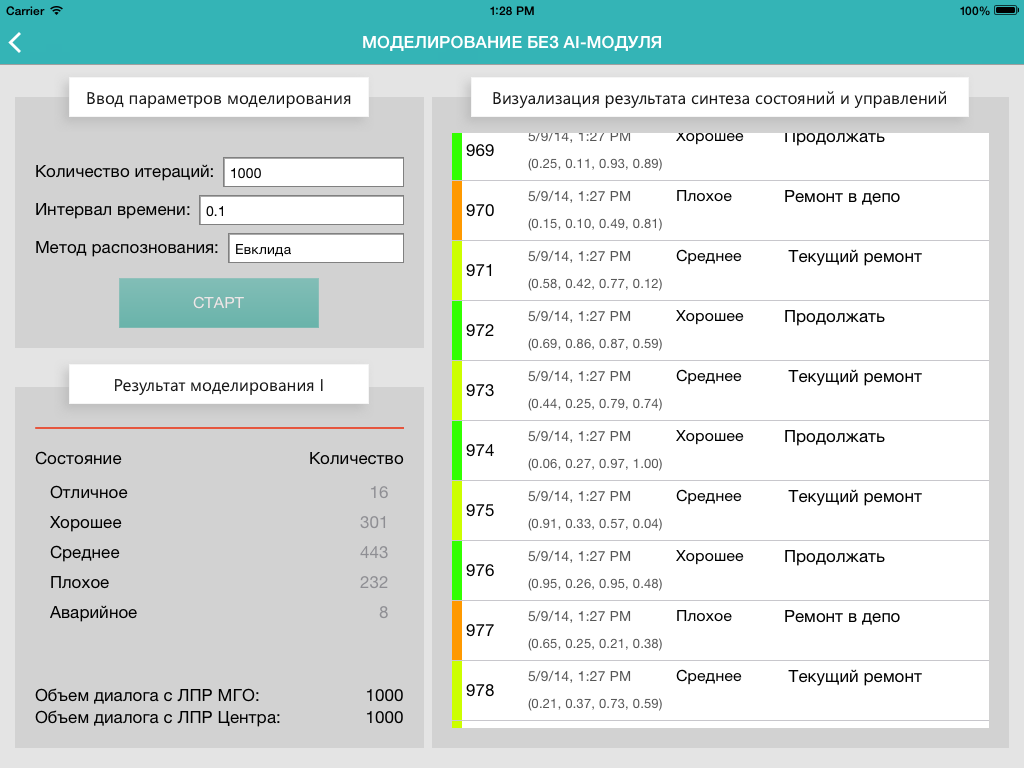


**Рисунок 3.26 – Сохранение модели в БД**

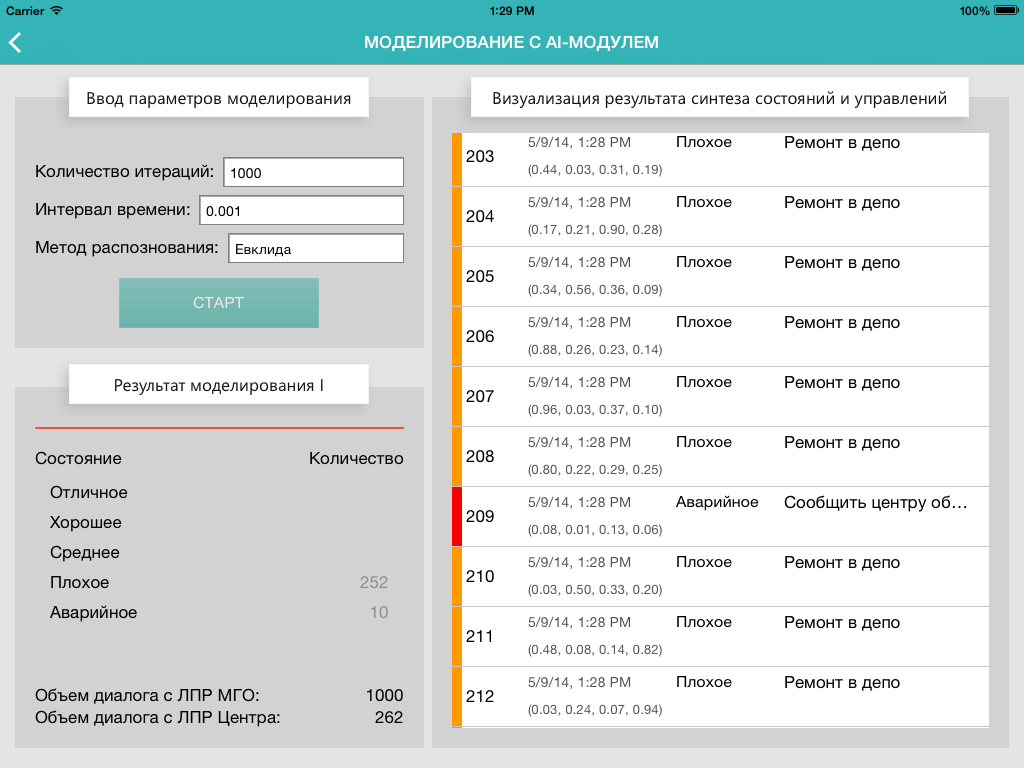
**Шаг 4.** Ввод количества итераций, интервала времени, а также метода распознавания (рисунок 3.27). Моделирование состояния объекта в диагностических точках без использования AI-модуля (рисунок 3.28) и с использованием AI-модуля (рисунок 3.29).



**Рисунок 3.27 – Ввод параметров моделирования**



**Рисунок 3.28 – Результат моделирования без AI-модуля**



**Рисунок 3.29 – Результат моделирования с AI-модулем**

В результате выполнения алгоритмов на экран выводится вся необходимая информация для ЛПР центра и ЛПР объекта наблюдения.

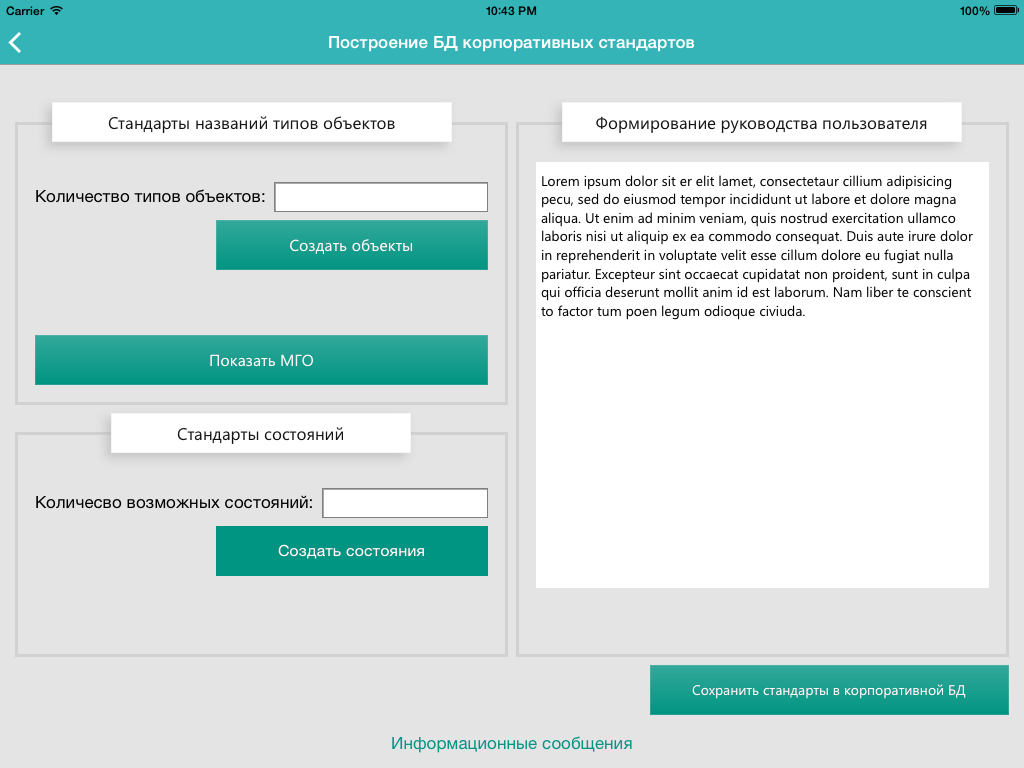
**Шаг 5.** Сравнение результатов моделирования без AI-модулем и с AI-модулем (рисунок 3.30).



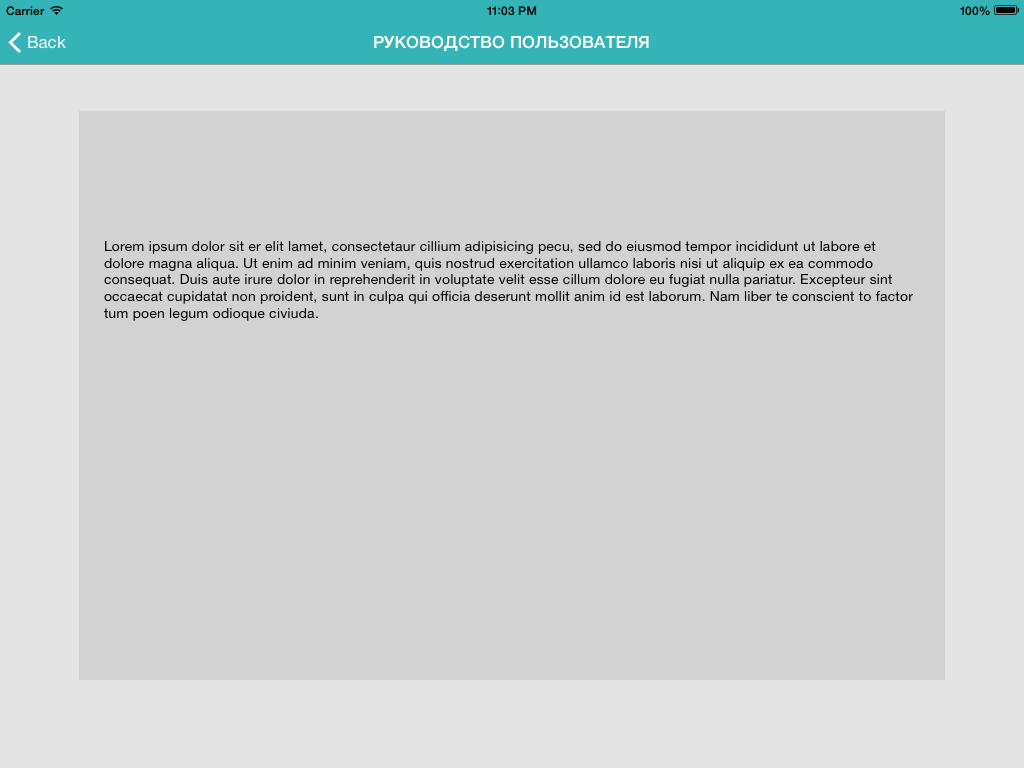
**Рисунок 3.30 – Сравнение результатов моделирования**

**Дополнительная функциональность.** Для удобства - можно ввести руководство пользователя (рисунок 3.31) и использовать его из главного меню (рисунок 3.32).

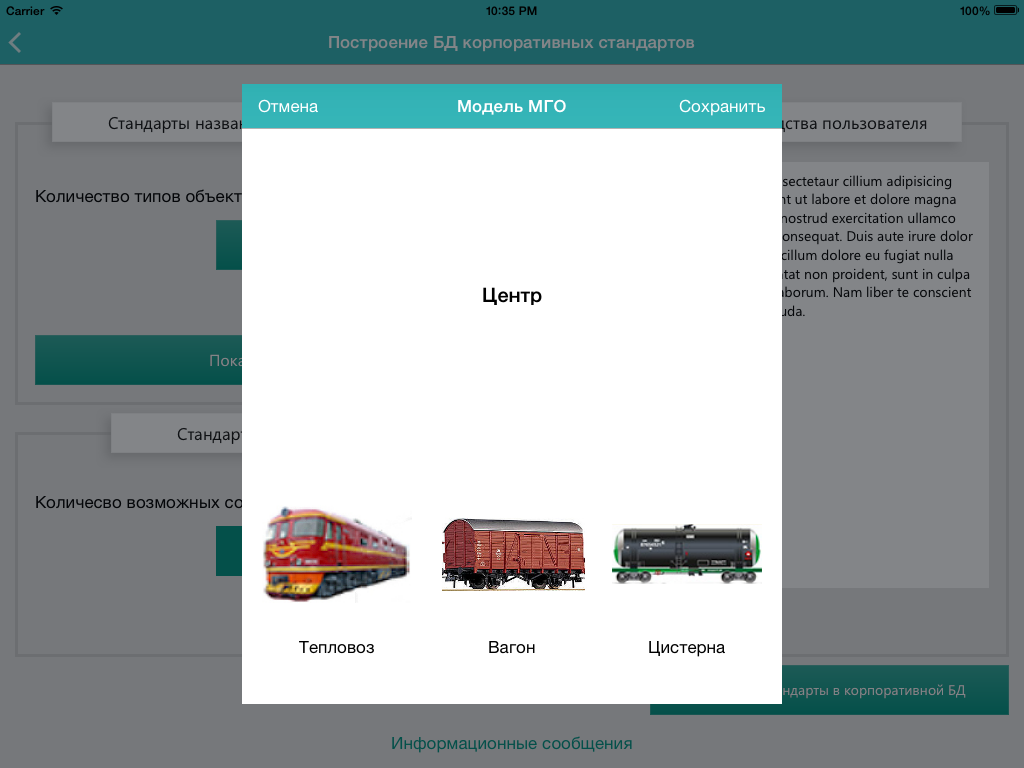
Существует возможность построить визуализацию сцены (рисунок 3.33).

****

**Рисунок 3.31 – Формирование руководства пользователя**

****

**Рисунок 3.32 – Руководства пользователя**



**Рисунок 3.33 – Визуализация сцены**

**3.11 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 OmniGraffle разработан комплект эскизов интерфейсов, соответствующих архитектуре;
* средствами языка Objective-C разработана библиотека программ, реализующих архитектуру, на ее основе построена система Monitoring;
* проведена апробация системы на примере моделирования процесса организации мониторинга движения железнодорожного состава.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения преддипломной практики были получены следующие основные результаты:

* изучена литература по мониторингу, выявлена слабо изученная проблема мониторинга структурированных мобильных объектов;
* сформулирована задача на решение проблемы;
* построены модели сцены мониторинга, включая центр и объект наблюдения;
* разработаны алгоритмы синтеза состояния и соответствующего управляющего решения для объекта;
* изучены операционная система Mac OS и средства разработки программного обеспечения Xcode, Objectice-C, Cocoa;
* разработано программное обеспечение для автоматизации мониторинга структурированных мобильных объектов.

Основной результат заключается в формировании цельного взгляда на мониторинг, обобщающий множество частных решений.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Аэрокосмический мониторинг объектов нефтегазового комплекса // Под редакцией Бондура В.Г. – М.: Научный мир, 2012. – 558 с.
2. Горячкин, В.В.,Демеш Н.Н., Коротаев Н.А. Методические указания по оформлению и защите курсовых, дипломных работ и других отчетных документов студентов университета. **–** Мн.: БГУ, 2005. **– 50 с.**
3. Израэль Ю.А. Экология и контроль состояния природной среды. – Л.: Гидрометеоиздат, 1979. – 376 с.
4. Малышев М.Л. Мониторинг социально-трудовой сферы. – Перспектива, 2007. – 280 с.
5. Смирнов С.В. Онтологическое моделирование в ситуационном управлении // Онтология проектирования. – 2012. – № 2(4). – С. 16-24.
6. Тоценко, В.Г. Методы и системы поддержки принятия решений. – Киев: Наукова думка, 2002. – 342 с.
7. Уилсон Э. Мониторинг и анализ сетей. – М.:Лори,2002.– 368 с.
8. Хиллегасс А. Objective-C. Программирование для iOS. – Спб, Питер, 2012. – 253 с.