| **№** | **Краткое описание** | **Речь** |
| --- | --- | --- |
| 1 | Тема, ФИО | Уважаемые члены Государственной Аттестационной комиссии, хочу продолжить вашему вниманию вторую часть дипломной работы, первую часть которой только что представил мой коллега. |
| 2 | Консоль | В первой части было представлено ядро системы. Оно по идеологическим причинам работает в консольном режиме. Этот режим очень удобен для запуска и функционирования, но при этом он не очень удобен для восприятия человеком. Ведь в человеческом мозгу нет такой единицы информации как слово. Мозг все запоминает образами, в большинстве своем визуальными. ~~Конечно, в тексте можно уместить больше информации на единицу занимаемой площади.~~ |
| 3 | Конфигурационные файлы | Для управления консольными приложениями часто используют конфигурационные файлы. Но у этого способа есть ряд недостатков. Конечно, некоторые из этих недостатков можно решить дополнительными средствами.(Разве консоль управления не является доп. средством?) Например решить первую проблему можно сделав доступным один файл конфигурации с удаленного сервера. Это возможно… ВОЗМОЖНО решит одну проблему, но возникнет ряд других (каких-то) неудобств. |
| 4 | Структура проекта -  страшная схема | Для решения этих проблем (и не только)(а каких еще?) и необходима вторая часть дипломной работы. Само по себе ядро может проделать большую работу, но для того, чтобы получить выгоду от этой работы, необходимо еще и эффективно использовать результаты этой работы. **Панель управления** – инструмент управления работой исполняемой среды и визуализации информации процесса и результатов этой работы. **Прикладной интерфейс программирования** позволяет разрабатывать модули мониторинга на основе унифицированного каркаса исходного кода модуля. В текущей реализации интерфейс программирования модулей представляется каркасом с одним публичным методом – «invoke(..)» (рисунок 3.19). Параметры модуля мониторинга могу передаваться как обычная коллекция или список языка Python. |
| 5 | Модель приложения MVC | В основу модели приложения был положен шаблон проектирования, называемый MVC или Model-View-Controller. Основная идея этого шаблона – разделение приложения на три (у вас только три?) основных компонента с «слабой» связанностью между ними. Компонент Модель реализован классом домен, который содержит информацию о ядре и логику получения, обработки и предоставления этой информации. К компоненту Представление или интерфейс можно отнести класс Главное окно, которое является пользовательским интерфейсом в классическом понимании этого выражения. Управляющий компонентом Поведение является класс координатор, который согласовывает информацию из домена и ее отображение в  пользовательском интерфейсе. |
| 6 | Диаграмма последовательностей | Сейчас Вы видите общую схему функционирования Панели управления и ее взаимодействия с ядром. Ядро через свой драйвер Discoverer оповещает панель о своей активности через определенные промежутки времени. Так же с помощью этого вызова передается минимальная информация о ядре. Так называемый контекст ядра. На основе которого делается вывод об изменение ядра. Если изменение произошло, то домен обновляет локальную информацию и сообщает об этом интерфейсу, который обновляет отображение. |
| 7 | Архитектура приложения  диаграмма классов |  |
| 8 | Хранение информации  Домен |  |
| 9 | Взаимодействие компонентов –  Координатор |  |
| 10 | Визуализация –  Главное окно |  |
| 11 | Использование API  кусок кода |  |
| 12 | Развертывание модуля  3 скриншота |  |
| 13 | Итоги |  |
| 14 | Спасибо! Вопросы? |  |