M-Smart OS  
  
Описание платформы

Оглавление

[Общие принципы. Задачи и архитектура 2](#_Toc321814321)

[Установка и настройка. Системные требования 3](#_Toc321814322)

[Иерархия сервисов 3](#_Toc321814323)

[Структура папок и файлов shipyard 3](#_Toc321814324)

[Описатели сборок 4](#_Toc321814325)

[Репозиторий-архив кода 5](#_Toc321814326)

[Рабочий репозиторий кода 5](#_Toc321814327)

[Релиз-репозиторий кода 5](#_Toc321814328)

[Процесс трансляции и синтеза сборок 5](#_Toc321814329)

[Публикация отладочной версии 5](#_Toc321814330)

[Архитектура сервиса 5](#_Toc321814331)

[Разработка сервисов. Язык МАЯ-ГРАФИТ 6](#_Toc321814332)

[Пустой сервис. Описатель 6](#_Toc321814333)

[Пустой сервис. Алгоритмический файл 6](#_Toc321814334)

[Вершины. Обработка сообщений 7](#_Toc321814335)

[Стандартные типы вершин. Синтез новых вершин 8](#_Toc321814336)

[Имитация наследования классов 8](#_Toc321814337)

[Синтаксические конструкции 8](#_Toc321814338)

[Блок cmd() 8](#_Toc321814339)

[Передача сообщения. msg() 8](#_Toc321814340)

[Работа с переменными. setglobal(), setlocal() 9](#_Toc321814341)

[Условия. if() 9](#_Toc321814342)

[Циклы. cycle(), through() 9](#_Toc321814343)

[Работа с полями объектов. Конструирование объектов. getfield(), setfield(), record(), field() 9](#_Toc321814344)

[Трансляция с использованием универсального процессора алгоритмов 9](#_Toc321814345)

[Разработка сервисов. Язык описания графических интерфейсов 9](#_Toc321814346)

[Элементы управления 9](#_Toc321814347)

[Простой пример разработки графического интерфейса 10](#_Toc321814348)

[Использование списков, флажков и переключателей 11](#_Toc321814349)

[Стандартные компоновки интерфейса 11](#_Toc321814350)

[«Девятка» 11](#_Toc321814351)

[«Шестерка» 11](#_Toc321814352)

[«Окно с заголовком – тройка» 11](#_Toc321814353)

[Панель инструментов 11](#_Toc321814354)

[Скроллер 11](#_Toc321814355)

[Веб-браузер 11](#_Toc321814356)

[Реализация меню и панелей инструментов 11](#_Toc321814357)

[Дизайн с применением CSS-стилей. «Перекраска» 11](#_Toc321814358)

[Локализация приложений. Стилистическая локализация 11](#_Toc321814359)

[Работа с локальной файловой системой приложения 11](#_Toc321814360)

[Работа с локальной файловой системой устройства 11](#_Toc321814361)

[Разработка веб-сервисов. Шлюзование на РНР через mongoose 11](#_Toc321814362)

[Разработка веб-сервисов. Система mongoose. Язык программирования М10. 12](#_Toc321814363)

[Обработка 12](#_Toc321814364)

[Работа с мультимедиа 12](#_Toc321814365)

[Работа с СМС 12](#_Toc321814366)

[Разработка сервисов. Библиотечный граф ядра 12](#_Toc321814367)

[Разработка сервисов. Библиотечный пул пользовательского уровня 12](#_Toc321814368)

[Сервисы с логином и паролем 12](#_Toc321814369)

[Идентификационная карточка 12](#_Toc321814370)

[Сервис-фасад для запуска других сервисов 12](#_Toc321814371)

[Новостные ленты 12](#_Toc321814372)

[Электронные формы 12](#_Toc321814373)

[Поисковый движок 13](#_Toc321814374)

# Общие принципы. Задачи и архитектура

Платформа M-Smart призвана обеспечить абстрагированный интерфейс, моделирующий типовое мобильное устройство, с возможностями, достаточными для построения информационных систем в бизнес-сегменте.

Технологически обеспечивается поддержка подмножества возможностей базовой платформы – J2ME. Исключаются возможности J2ME, предусматривающие взаимодействие программы с устройством.

С точки зрения программиста, платформа представляет мобильное устройство в составе таких узлов как:

* Дисплей
* Клавиатура
* Дисплейный сенсор
* Внутренняя файловая система (доступна только самой программе)
* Внешняя файловая система устройства (карты памяти и т. д.)
* Доступ к интернет-подключению
* SMS-интерфейс

Собранные мобильные приложения на базе M-Smart готовы к исполнению на любой J2ME-системе. Шлюз для Android позволяет получить версию в формате файла APK и запустить на соответствующем устройстве.

Ключевые понятия в M-Smart – сервис и сборка. Сервис соответствует понятию программы, сборка – близко к понятию дистрибутива в мире Linux. Физически сборка является мобильным приложением в формате jar.

Система shipyard – система управления построением сборок в M-Smart. Функционал системы включает в себя:

1. Управление репозитарием кода
2. Управление локализацией программного обеспечения
3. Управление версиями
4. Компоновку сборок
5. Компиляцию сборок.

Планируется интеграция функций:

1. Публикация сервисов в публичном репозитории
2. Публикация сборок
3. Автоматическая сборка anroid-версий

Shipyard управляет одновременно всеми текущими проектами и сборками. Ввиду этого, при командной работе необходимо размещать «центральный» SDK на сервере, а на локальных компьютерах разработчиков – «опытные станции», код из которых при завершении отладки копируется на центральный сервер.

# Установка и настройка. Системные требования

# Иерархия сервисов

Сервисы в системе M-Smart архитектурно идентичны программам в операционной системе.

Два базовых системных сервиса – системное ядро (system) и рабочий стол (desktop) – являются фундаментальными и включаются в любую сборку. Данные сервисы включают стандартную базовую функциональность.

Поверх базовых сервисов надстраиваются инфраструктуры – функциональные разделы системы. К примеру, инфраструктурами являются системы MOBITESTUM, SARS, INFORMER. Поверх инфраструктур строятся непосредственно конечные пользовательские сервисы, которые используют возможности инфраструктур. К примеру, на базе MOBITESTUM построен тренажер MOBITESTUM T и система интерактивных викторин MOBITESTUM V.

Иерархия сервисов определяет конвенции их именования. Внеконвенциональные сервисы – system и desktop. Дальнейшая иерархия подчиняется конвенциям именования сервисов.

Имя конвенционального сервиса состоит из двух слогов, разбитых знаком «\_» - префикса и имени сервиса. Например, egov\_s\_lav - префикс egov, имя сервиса s\_lav. Это означает, что данный сервис является подчиненным к сервису egov, который в свою очередь имеет идентификатор sys\_e\_gov – т. е. является системной инфраструктурой. Префикс sys – определяющий идентификатор системной инфраструктуры.

# Структура папок и файлов shipyard

|  |  |
| --- | --- |
| assemblies | Собранный сборки M-Smart. Содержимое папки генерируется автоматически |
| **assembly-configs** | XML-конфигурации всех сборок |
| **built-assemblies** | Построенные сборки |
| debug-classes | Классы для построения отладочной сборки. Системная папка, модификация содержимого приводит к неработоспособности системы. |
| desktop-classes | Классы для построения настольной сборки. Системная папка, модификация содержимого приводит к неработоспособности системы. |
| **development** | Репозитарий исходного кода, который находится в разработке. |
| mobile-classes | Классы для построения мобильной сборки. Системная папка, модификация содержимого приводит к неработоспособности системы. |
| **release** | Репозитарий исходного кода, который готов к релизу. |
| **source** | Архив исходного кода, размещаемый в папке development. Используется для удобства работы. Внутри данной папки находятся подпапки, содержимое которых при трансляции размещается в папке development. |
| **deploy.xml** | Файл, который задает размещение  <deployment  KUIXRES="C:\Users\Mykola Kotyuck\Desktop\kuix\res\"  DESKTOP="C:\Program Files\M-Research\M-Smart Desktop Edition\"  DEFAULT="uoc"  />  KUIXRES – папка, используемая для размещения построенной сборки с целью запуска из NetBeans  DESKTOP – папка, используемая для публикации собранной сборки для просмотра в отладочном эмуляторе.  DEFAULT – имя сборки, которая публикуется в папках KUIXRES и DESKTOP. |
| **shipyard.exe** | Основной рабочий файл системы, выполняющий автоматическое построение всех заданных сборок. Не предусматривает параметров командной строки. |

# Описатели сборок

Описатель сборки – это файл в формате XML, описывающий состав и конфигурацию конкретной сборки.

<assembly REPOSITORY="development" LANGUAGE="uawhite" SBG="white" MASTER="sys\_s\_ars" DESKTOP="sys\_s\_ars" SERVER="mplatforma.com:81">

<services>

<system/>

<desktop/>

<sys\_m\_ts/>

<mts\_t/>

<mts\_r/>

<mts\_c/>

<sys\_i\_nfo/>

<info\_news/>

<sys\_s\_ars/>

<sars\_int/>

<sars\_send/>

<sars\_store/>

<sars\_post/>

<sars\_arch/>

<sars\_dnld/>

<sars\_load/>

<sars\_admin/>

<sars\_eid/>

<sars\_debug/>

<sars\_cati/>

<sars\_tasks/>

</services>

<top4>

<sys\_s\_ars/>

<sars\_int/>

<sars\_post/>

</top4>

</assembly>

Параметры описателя сборки:

* REPOSITORY – репозиторий кода, на базе которого строится сборка. Разновидность репозитория – development или release.
* LANGUAGE – язык (локализация) сборки. Более подробно см. в разделе «локализация».
* SBG – структура подложки. Вариант white – по умолчанию.
* MASTER – главный сервис, определяющий логотип сборки.
* DESKTOP – сервис «рабочего стола», т. е. первый сервис, который запускается после старта системы.
* SERVER – адрес сервера в сети, с которым выполняется взаимодействие сборки.

Вложенный тег services задает список сервисов, которые включаются в сборку.

Вложенный тег top4 задает список сервисов, которые доступны изначально через рабочий стол системы.

# Репозиторий-архив кода

# Рабочий репозиторий кода

# Релиз-репозиторий кода

# Процесс трансляции и синтеза сборок

# Публикация отладочной версии

# Архитектура сервиса

Сервис в M-Smart инкапсулирован в папку, имя которой и является именем сервиса, включает в себя такие компоненты как:

* Файл-описатель сервиса (имя файла ss.sd)
* Семейство файлов с логотипом сервиса (sl\*\*.cpp)
* Исполняемый файл service.m – файл, определяющий алгоритм работы сервиса.

В целом, в M-Smart распространены такие форматы файлов:

* \*.m – файлы скомпилированного промежуточного кода
* \*.w – файлы описания графического интерфейса
* \*.cpp – обработанные графические изображения в формате png
* \*.sd – дескрипторы сервисов

В отношении файлов \*.cpp важно понимать, что графические изображения в системе M-Smart адаптируются к 4-м типовым стандартным габаритам, с кодовыми наименованиями 30, 40, 60, 80. Таким образом, каждая картинка представлена в четырех версиях с различным разрешением. Для выбора нужной картинки предусмотрен внутрисистемный протокол **mlsp**.

Размещение других файлов внутри папки сервиса произвольно и определяется разработчиком.

# Разработка сервисов. Язык МАЯ-ГРАФИТ

## Пустой сервис. Описатель

Представим, что перед нами стоит задача написать новый сервис. Для этого необходимо:

* 1. создать для сервиса папку в репозитории development
  2. создать описатель сервиса
  3. разместить изображение-логотип
  4. создать и скомпилировать файл логики
  5. создать описатель сборки, включающей наш сервис

В качестве рабочего имени сервиса будем использовать sys\_demo. Соответственно, описатель нашей сборки будет иметь вид:

<assembly REPOSITORY="development" LANGUAGE="ua" SBG="white" MASTER="system"

DESKTOP="desktop" SERVER="mplatforma.com:81">

<services>

<system/>

<desktop/>

<sys\_demo/>

</services>

<top4>

<sys\_demo/>

</top4>

</assembly>

В директории нашого сервиса создадим файл ss.sd и заполним его следующим образом:

<service SHIFT="0" VERSION="10" EDITION="2" name ="sys\_demo" caption ="Демонcтраційний сервіс" logo="tile://sys\_demo/sl30.cpp" description="Опис сервісу"/>

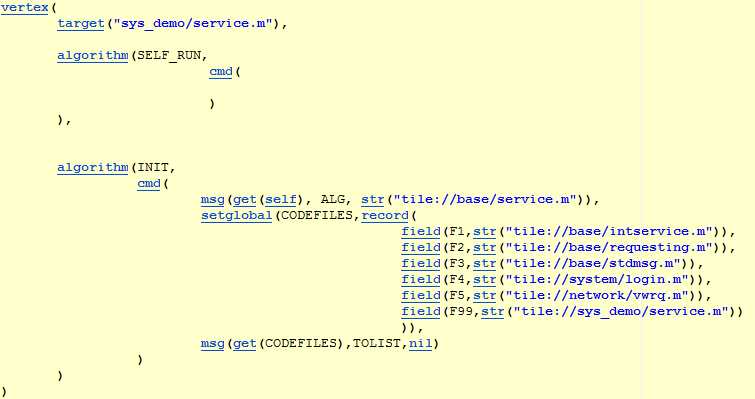
Формальная принадлежность сервиса к системным инфраструктурам (имя с префиксом sys) приводит к тому, что наш сервис будет по умолчанию доступен из главного меню системы. Соответственно, для «навешивания» логических функций необходимо описать алгоритмический файл сервиса на языке МАЯ-ГРАФИТ, а также создать при необходимости файлы с описанием графического интерфейса и предусмотреть команды визуализации соответствующих элементов графического интерфейса.

## Пустой сервис. Алгоритмический файл

Алгоритмический файл задает логику работы сервиса. Для начала важно отметить общую структуру стандартного алгоритмического файла, а именно синтаксис скобочного выражения, имеющий вид *вершина(вершина(вершина, , ,), вершина(вершина,вершина(,,,,)))*. Также важно рассмотреть непосредственно подсвечиваемые синтаксические конструкции, а именно **vertex, target и algorithm**.

Директива target задает место расположения целевого скомпилированного файла. В данном случае это файл service.m в папке нашего сервиса sys\_demo.

Алгоритм в языке МАЯ-ГРАФИТ – это аналог метода в объектно-ориентированных языках.



Важно отметить особенности алгоритма INIT, который обрабатывает стандартное конвенционное сообщение – т. е. выполняет функции конструктора. В языке МАЯ-ГРАФИТ важная задача конструктора – подключение унаследованных обработчиков сообщений. К примеру:

1. msg(get(self), ALG, str("tile://base/service.m")) – базовый класс сервиса
2. field(F1,str("tile://base/intservice.m")) – сервис с графическим интерфейсом
3. field(F2,str("tile://base/requesting.m")) – сервис стандартных окон-запросов
4. field(F3,str("tile://base/stdmsg.m")) – сервис стандартизованных окон-сообщений
5. field(F4,str("tile://system/login.m")) – сервис с поддержкой запроса на вход с логином и паролем
6. field(F5,str("tile://network/vwrq.m")) – сервис с поддержкой сетевых запросов (вызовов веб-методов).

## Вершины. Обработка сообщений

Фундаметальные понятия в системе МАЯ-ГРАФИТ - объектный граф и обмен сообщениями. Объектный граф – это структура из объектов, которая функционирует по принципу, близкому к функционированию объектов в языках Java / C++ / Delphi. Однако имеется ряд принципиальных различий.

Объектная вершина имеет ассоциативное хранилище данных, построенное по принципу «ключ – значение». Таким образом реализуется концепция полей в экземпляре класса. Объект может посылать сообщения другим объектам, при чем каждое сообщение имеет один параметр – вершину объектного графа, и идентификатор. Вершина может реализовать обработку сообщений.

Для того, чтобы вершина выполняла задачи обработки сообщений, необходимо загрузить в эту вершину алгоритмы-обработчики сообщений. Алгоритмы могут быть загружены исключительно извне, путем отсылки вершине системного сообщения ALG.

Для приведения в движение всей цепочки вершин и сообщений в системе существует понятие стартовой вершины – BOOT-вершины. Эта вершина создается извне, исполнительной средой системы. Затем выполняется отправка стартового сообщения данной вершине, и выполняется фактический запуск системы.

В случае нашего сервиса из вершины BOOT первоначально инициируется вызов методa INIT, а затем метода SELF\_RUN, который является собственно рабочей процедурой сервиса – неким аналогом функции main в программах на С++.

В дальнейшем проиллюстрируем работу с

## Стандартные типы вершин. Синтез новых вершин

В M-Smart предусмотрены следующие стандартные типы вершин объектного графа:

* Общая-числовая (graphit.me.Vertex)
* Строковая (graphit.me.str)
* Байтовый массив (graphit.me.arbyte)
* Календарь (graphit.me.Calendar)
* HTTP-соединение (graphit.me.HTTPConnectionVertex)
* Входящий поток (graphit.me.InputStreamVertex)
* Исходящий поток (graphit.me.OutputStreamVertex)
* Мультимедийный контроллер (graphit.me.Multimedia)
* Контроллер информационного носителя (graphit.me.MCControl)
* Контроллер SMS-сообщений (graphit.me.SMSSendVertex)

Новая вершина может быть получена с помощью:

1. Объявления литерала строки или целого числа (конструкции вида int(0) или str(“hello”))
2. Операции new вида new(graphit.me.SMSSendVertex).

## Имитация наследования классов

Логика поведения конкретной вершини в системе МАЯ-ГРАФИТ задается исключительно последовательностью загрузки \*.m файлов путем вызова сообщения ALG. Таким образом мы определяем то, как эта вершина будет реагировать на определенные сообщения.

Для того, чтобы повторно использовать реализованные для другой вершины алгоритмы, достаточно подгрузить файл \*.m этой вершины. При этом, наша вершина будет реагировать на сообщения аналогично вершине-донору. Алгоритмы можем перекрыть (реализовать отличным от изначальной вершины способом) внутри нашего файла, однако, в этом случае необходимо включать наш файл в цепочку наследования в качестве последнего файла в списке.

## Синтаксические конструкции

### Блок cmd()

Блок команды аналогичен фигурным скобкам в С++ или комбинации begin end; в Паскале. Заключает внутри себя последовательность операторов, фактически является одним оператором.

### Передача сообщения. msg()

Блок команды аналогичен фигурным скобкам в С++ или комбинации begin end; в Паскале. Заключает внутри себя последовательность операторов, фактически является одним оператором.

### Работа с переменными. setglobal(), setlocal()

Блок команды аналогичен фигурным скобкам в С++ или комбинации begin end; в Паскале. Заключает внутри себя последовательность операторов, фактически является одним оператором.

### Условия. if()

Блок команды аналогичен фигурным скобкам в С++ или комбинации begin end; в Паскале. Заключает внутри себя последовательность операторов, фактически является одним оператором.

### Циклы. cycle(), through()

Блок команды аналогичен фигурным скобкам в С++ или комбинации begin end; в Паскале. Заключает внутри себя последовательность операторов, фактически является одним оператором.

### Работа с полями объектов. Конструирование объектов. getfield(), setfield(), record(), field()

Блок команды аналогичен фигурным скобкам в С++ или комбинации begin end; в Паскале. Заключает внутри себя последовательность операторов, фактически является одним оператором.

## Трансляция с использованием универсального процессора алгоритмов

Для трансляции необходимо….

# Разработка сервисов. Язык описания графических интерфейсов

Подсистема графического интерфейса в M-Smart базируется на трех компонентах – стандартных инструментах описания на базе xml, таблицах стилей css и библиотеке готовых наработанных решений.

Ключевые понятия в M-Smart в отношении графического интерфейса:

* Сервис – собственно программа. Сервис управляет отображением экранных кадров и принимает сообщения на обработку, а также поставляет данные для визуализатора.
* Экранный кадр (screen) – информационная единица, что показывается на экране в один момент. Близко понятию окна в ОС Windows. Экранный кадр содержит элементы управления.
* Сообщение графического интерфейса передается при действии пользователя в экранном кадре и обрабатывается вершиной сервиса.
* Визуализатор данных – синтаксическая конструкция, которая позволяет экранному кадру динамически извлекать данные из сервиса и отображать их по ходу работы. Сервис должен самостоятельно инициировать отображение информации визуализатором.

## Элементы управления

В M-Smart поддерживаются следующие элементы управления:

* Контейнеры (структурирующие элементы, способные только содержать другие элементы управления)
* Текстовые метки
* Картинки
* Кнопки
* Флажки
* Переключатели
* Списки однотипных элементов
* Зоны прокрутки
* Индикаторы процесса
* Текстовые редактируемые поля
* HTML-вьюер.

## Простой пример разработки графического интерфейса

<screen class="m11scr" cleanupwhenremoved="true">

<container class="m11w">

<picture class="m11wp">

mlsp#//scale-img/sys\_rprt/sl

</picture>

<container class="m11wt">

<textarea class="m11wc">

Репортер

</textarea>

<textarea class="m11wi">

Текстове представлення

</textarea>

</container>

</container>

<scrollpane class="m11mc">

<textarea style="font-size:small;" styled="true">

@{TEXTVIEW}

</textarea>

</scrollpane>

<container class="m11bottombar">

<container class="m116mscc">

<\_style>

gap:${gapsz} ${gapsz};

</\_style>

<button class="gridmenu" enabled="false">

<\_style>

bg-image:url(mlsp#//scale-img/base/lpempty);

</\_style>

</button>

<button class="gridmenu" enabled="false">

<\_style>

bg-image:url(mlsp#//scale-img/base/lpempty);

</\_style>

</button>

<button class="gridmenu" onaction="PROGRESS#RETURN\_BACK" shortcuts="9|back|delete|softright">

<\_style>

bg-image:url(mlsp#//scale-img/base/lpcancel);

min-size:${sz} ${halfsz};

</\_style>

</button>

</container>

</container>

</screen>

## Использование списков, флажков и переключателей

## Стандартные компоновки интерфейса

### «Девятка»

### «Шестерка»

### «Окно с заголовком – тройка»

### Панель инструментов

### Скроллер

### Веб-браузер

## Реализация меню и панелей инструментов

## Дизайн с применением CSS-стилей. «Перекраска»

# Локализация приложений. Стилистическая локализация

# Работа с локальной файловой системой приложения

# Работа с локальной файловой системой устройства

# Разработка веб-сервисов. Шлюзование на РНР через mongoose

# Разработка веб-сервисов. Система mongoose. Язык программирования М10.

# Обработка

# Работа с мультимедиа

# Работа с СМС

# Разработка сервисов. Библиотечный граф ядра

# Разработка сервисов. Библиотечный пул пользовательского уровня

## Сервисы с логином и паролем

## Идентификационная карточка

## Сервис-фасад для запуска других сервисов

## Новостные ленты

## Электронные формы

## Поисковый движок

## Сервисы с кешированием информации из сети