Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Южно-Уральский государственный университет»

(национальный исследовательский университет)

Институт естественных и точных наук

Факультет Математики, механики и компьютерных наук

Кафедра математического и компьютерного моделирования

Управление программными продуктами

ЮУрГУ– 010400**.**68.2017.049.001.ПЗ ПР.

|  |  |  |
| --- | --- | --- |
|  |  | Руководитель,  к.т.н., доцент Дударева В.И.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. |
|  |  |  |
|  |  | Автор работы  Студент группы ЕТ-224  В.А. Безбородов  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. |
|  |  |  |
|  |  | Работа защищена с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. |

Челябинск, 2017

Оглавление

[Оглавление 2](#_Toc480492123)

[1 Интернет вещей 3](#_Toc480492124)

[1.1 Описание работы программы 3](#_Toc480492125)

[1.2 Комментарии по модели 6](#_Toc480492126)

[2 Технология RAD быстрой разработки приложений 7](#_Toc480492127)

[2.1 Введение 7](#_Toc480492128)

[2.2 Назначение 7](#_Toc480492129)

[2.3 Применение 8](#_Toc480492130)

[2.4 Фазы разработки 10](#_Toc480492131)

[2.5 Преимущества 12](#_Toc480492132)

[3 Проект по реконструкции стадиона 13](#_Toc480492133)

[3.1 Участники проекта 13](#_Toc480492134)

[3.2 Устав проекта 14](#_Toc480492135)

[3.3 Риски проекта 14](#_Toc480492136)

[3.4 График проекта 15](#_Toc480492137)

[3.5 Бюджет проекта 16](#_Toc480492138)

[4 Унифицированный язык моделирования UML 17](#_Toc480492139)

[Список литературы 19](#_Toc480492140)

# Интернет вещей

## Описание работы программы

Если тело бросить под углом к горизонту, то в полете на него действуют сила тяжести и сила сопротивления воздуха. Если силой сопротивления пренебречь, то остается единственная сила – сила тяжести. Поэтому вследствие 2-го закона Ньютона тело движется с ускорением, равным ускорению свободного падения .

Введем в программный код константу, представляющую ускорение свободного падения.

var g = 9.80665; // ускорение свободного падения, м / с^2

Проекции ускорения на координатные оси равны:



Проекции скорости тела, следовательно, изменяются со временем следующим образом:



где  - начальная скорость,  - угол бросания.

По закону сохранения импульса, до взаимодействия:



где  - импульсы клюшки и шарика до взаимодействия соответственно,  - массы клюшки и шарика,  - скорости клюшки и шарика до взаимодействия.

В программном коде зададим константы для значений масс:

var m1 = 1.5; // масса клюшки, кг

var m2 = 0.1; // масса шарика, кг

После взаимодействия характеристики объектов изменятся следующим образом:



где  - импульсы клюшки и шарика после взаимодействия соответственно,  - массы клюшки и шарика,  - скорости клюшки и шарика после взаимодействия.

Согласно 3-му закону Ньютона, , следовательно:





Поскольку начальная скорость шарика  (шарик покоится), то



Скорость шарика после взаимодействия (удара):



Скорость клюшки до и после удара можно определить, если отнести ширину  датчика ко времени прохода клюшки над датчиком.

Зададим в программном коде константу, определяющую ширину датчика.

var S = 0.01; // ширина датчика, м

Время пролета клюшки над первым и вторым датчиками можно получить, зная разницу времени их срабатывания. Время срабатывания хранится в property вещи.

var t0 = me.h1 - me.l1; // время пролета клюшки над первым датчиком, с

var t1 = me.h2 - me.l2; // время пролета клюшки над вторым датчиком, с

Рассчитаем скорость клюшки до и после удара.

var v01 = S / t0; // скорость клюшки до удара, м / с

var v1 = S / t1; // скорость клюшки после удара, м / с

Тогда начальная скорость мяча будет определяться следующим соотношением.

var v0 = m1 \* (v01 - v1) / m2; // м / с

Положим угол бросания (константа как характеристика определенной клюшки).

var alpha = Math.PI / 4; // радиан

Координаты тела, следовательно, изменяются так:



Если принять за начало координат вторую линию положение второго датчика, то , . Тогда:



В момент приземления координата . Отсюда получаем для времени полета:



Второе значение времени, при котором высота равна нулю, равно нулю, что соответствует моменту бросания.

Дальность полета - это значение координаты  в конце полета, т.е. в момент времени, равный :



Дальность полета мяча в метрах в программном коде присвоим выходной переменной.

result = Math.pow(v0, 2) \* Math.sin(2 \* alpha) / g;

## Комментарии по модели

Разумеется, в рассматриваемой модели механика реального удара при игре в гольф сильно упрощена. В модели принимаются следующие допущения.

* Существует только по одному датчику в первой и второй линиях, т.е. удар может производиться только по прямой.
* После приземления мяч не отскакивает (абсолютно неупругий удар).
* Датчики посылают серверу время начала и окончания пролета клюшки, данные сохраняются в property объекта thing.
* Все пересылаемые данные исключительно корректные (например, время начала пролета меньше времени окончания и т.п.), не существует потерь при передачи по каналу связи.

Такая модель имеет низкую конкурентоспособность на рынке производителей устройств для эмуляции удара при игре в гольф.

Модель рекомендуется использовать в учебных целях для освоения и закрепления знаний о физических законах взаимодействия двух тел.

# Технология RAD быстрой разработки приложений

## Введение

RAD (от англ. rapid application development — быстрая разработка приложений) — концепция создания средств разработки программных продуктов, уделяющая особое внимание быстроте и удобству программирования, созданию технологического процесса, позволяющего программисту максимально быстро создавать компьютерные программы. Практическое определение: RAD — это жизненный цикл процесса проектирования, созданный для достижения более высокой скорости разработки и качества ПО, чем это возможно при традиционном подходе к проектированию. С конца XX века RAD получила широкое распространение и одобрение. Концепцию RAD также часто связывают с концепцией визуального программирования.

Концепция RAD стала ответом на неуклюжие методы разработки программ 1970-х и начала 1980-х годов, такие как «модель водопада» (англ. Waterfall model). Эти методы предусматривали настолько медленный процесс создания программы, что зачастую даже требования к программе успевали измениться до окончания разработки. Основателем RAD считается сотрудник IBM Джеймс Мартин, который в 1980-х годах сформулировал основные принципы RAD, основываясь на идеях Барри Бойма и Скотта Шульца. А в 1991 году Мартин опубликовал известную книгу, в которой детально изложил концепцию RAD и возможности её применения. В настоящее время RAD становится общепринятой схемой для создания средств разработки программных продуктов.

## Назначение

RAD предполагает, что разработка ПО осуществляется небольшой командой разработчиков за срок порядка трёх-четырёх месяцев путём использования инкрементного прототипирования с применением инструментальных средств визуального моделирования и разработки. Технология RAD предусматривает активное привлечение заказчика уже на ранних стадиях — обследование организации, выработка требований к системе. Последнее из указанных свойств подразумевает полное выполнение требований заказчика как функциональных, так и нефункциональных, с учётом их возможных изменений в период разработки системы, а также получение качественной документации, обеспечивающей удобство эксплуатации и сопровождения системы. Это означает, что дополнительные затраты на сопровождение сразу после поставки будут значительно меньше. Таким образом, полное время от начала разработки до получения приемлемого продукта при использовании этого метода значительно сокращается.

## Применение

Технологию RAD целесообразно применять, когда четко определены некоторые приоритетные направления разработки проекта.

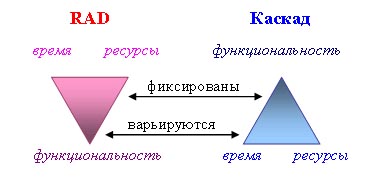


Рисунок 1 – Сравнение RAD и Каскадного метода

*Необходимо выполнение проекта в сжатые сроки.* Быстрое выполнение проекта позволяет создать систему, отвечающую требованиям сегодняшнего дня. Если система проектируется долго, то весьма высока вероятность, что за это время существенно изменятся фундаментальные положения, регламентирующие деятельность организации, то есть, система морально устареет ещё до завершения её проектирования.

*Нечетко определены требования к ПО.* В большинстве случаев заказчик весьма приблизительно представляет себе работу будущего программного продукта и не может четко сформулировать все требования к ПО. Требования могут быть вообще не определены к началу проекта либо могут изменяться по ходу его выполнения.

*Проект выполняется в условиях ограниченности бюджета.* Разработка ведётся небольшими RAD-группами в короткие сроки, что обеспечивает минимум трудозатрат и позволяет вписаться в бюджетные ограничения.

*Интерфейс пользователя (GUI) есть главный фактор.* Нет смысла заставлять пользователя рисовать картинки. RAD-технология дает возможность продемонстрировать интерфейс в прототипе, причём достаточно скоро после начала проекта.

*Возможно разбиение проекта на функциональные компоненты.* Если предполагаемая система велика, необходимо, чтобы её можно было разбить на мелкие части, каждая из которых обладает четкой функциональностью. Они могут выпускаться последовательно или параллельно (в последнем случае привлекается несколько RAD-групп).

RAD-технология не является универсальной, то есть её применение целесообразно не всегда. Например, в проектах, где требования к программному продукту четко определены и не должны меняться, вовлечение заказчика в процесс разработки не требуется и более эффективной может быть иерархическая разработка (каскадный метод). То же касается проектов, ПО, сложность которых определяется необходимостью реализации сложных алгоритмов, а роль и объём пользовательского интерфейса невелик.



Рисунок 2 – Основные принципы

Принципы RAD технологии направлены на обеспечение трёх основных её преимуществ — высокой скорости разработки, низкой стоимости и высокого качества. Достигнуть высокого качества программного продукта весьма непросто и одна из главных причин возникающих трудностей заключается в том, что разработчик и заказчик видят предмет разработки (ПО) по-разному.

* Инструментарий должен быть нацелен на минимизацию времени разработки.
* Создание прототипа для уточнения требований заказчика.
* Цикличность разработки: каждая новая версия продукта основывается на оценке результата работы предыдущей версии заказчиком.
* Минимизация времени разработки версии, за счёт переноса уже готовых модулей и добавления функциональности в новую версию.
* Команда разработчиков должна тесно сотрудничать, каждый участник должен быть готов выполнять несколько обязанностей.
* Управление проектом должно минимизировать длительность цикла разработки.

Принципы RAD применяются не только при реализации, но и распространяются на все этапы жизненного цикла, в частности на этап обследования организации, построения требований, анализ и дизайн.

## Фазы разработки

Планирование — совокупность требований, полученных при системном планировании и анализе процедуры разработки жизненного цикла (SDLC). На этом этапе пользователи, менеджеры и IT-специалисты обсуждают задачи проекта, его объём, системные требования, а также сложности, которые могут возникнуть при разработке. Фаза завершается согласованием ключевых моментов с RAD-группой и получением от руководителей проекта разрешения на продолжение.



Рисунок 3 – Модель быстрой разработки приложений (RAD)

Пользовательское проектирование — на протяжении данного этапа пользователи, взаимодействуя с системными аналитиками, разрабатывают модели и прототипы, которые включают в себя все необходимые системные функции. Для перевода пользовательских прототипов в рабочие модели RAD-группа обычно использует технику объединенной разработки приложений (JAD) и CASE-инструменты. Пользовательское проектирование оказывается длительным интерактивным процессом, который позволяет пользователям понять, изменить и в конечном счёте выбрать рабочую модель, отвечающую их требованиям.

Конструирование — этап, в котором основная задача заключается в разработке программ и приложений. Аналогична стадии «реализация» в SDLC. В RAD, однако, пользователи продолжают принимать участие и по-прежнему могут предлагать изменения или улучшения в виде разработанных ими докладов. В их задачи входит программирование и разработка приложений, написание кода, интеграция модулей и системное тестирование.

Переключение — включает в себя операции по конверсии данных, тестирование, переход на новую систему и тренировку пользователей. По своим задачам напоминает финальную стадию SDLC. Сравнивая с традиционными методами разработки ПО, весь процесс оказывается сжатым по времени. Как результат, новая система оказывается быстрее построенной, доставленной до заказчика и установленной на рабочих местах.

## Преимущества

Технология быстрой разработки приложений (RAD) позволяет обеспечить:

* быстроту продвижения программного продукта на рынок;
* интерфейс, устраивающий пользователя;
* лёгкую адаптируемость проекта к изменяющимся требованиям;
* простоту развития функциональности системы.

# Проект по реконструкции стадиона

Реконструкция территории парка при стадионе (пересечение Коммуны и Энгельса).

## Участники проекта

|  |  |
| --- | --- |
| **Заказчик** | Администрация г. Челябинска |
| **Спонсор** | * Муниципальное автономное учреждение «Центральный парк культуры и отдыха им. Ю.А.Гагарина» (директор учреждения: Шаехов Вадим Надимович). * Челябинская Детская Железная Дорога (Начальник: Токаренко Владимир Александрович). * УралГУФК (Ректор: Евгений Федорович Орехов). * Динамо, Всероссийское физкультурно-спортивное общество (Глава Попечительского совета: Владимир Колокольцев). * Пейнтбольный клуб «RUSH» (Директор пейнтбольного клуба: Юрий Недавний). |
| **Функциональные организации, участвующие в проекте** | * ООО ПКФ «Астра» - проектно-строительная компания г. Челябинска. * Питомник «КЕДР» - озеленение и благоустройство территории. Ландшафтные работы. |
| **Команда проекта** | PM, прораб, архитектор, бухгалтер,. |
| **Внешние участники** | Бригада наемных рабочих. |

## Устав проекта

|  |  |
| --- | --- |
| **Самое общее описание проекта** | Проект по реконструкции территории парка при стадионе (пересечение Коммуны и Энгельса) г. Челябинска. |
| **Цель или обоснование проекта** | Привлечение дополнительных инвестиций в развитие парка. Увеличение оборота существующих организаций, работающих на территории парка, за счет увеличения человеко-потока. |
| **Уровень полномочий менеджера проекта** | * Право вето. * Распределение полномочий между участниками. * Назначение сроков сдачи объекта. |
| **Допущения проекта** | * Проект имеет организационную поддержку со стороны руководства заказчика. * У организации-заказчика имеется возможность выделить персонал для обеспечения работ по проекту. |
| **Ограничения проекта** | * Внесение изменений в содержание проекта производится только с согласия всех сторон-участников. |
| **Бюджет проекта** | 5 млн. р. |

## Риски проекта

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Наименование риска** | **Вероятность наступления риска** | **Величина влияния риска на проект** | **Финансовый эффект от принятия риска** | **Стратегия реагирования** | **Описание реагирования на риск** | **Стоимость реагирования** |
| Необеспеченность кадрами | 0,15 | 1 | 2 500 000 | Уклонение | Увеличение штата | 100 000 |
| Изменения в технологии | 0,05 | 0,5 | 50 000 | Снижение | Новый техпроцесс | 30 000 |
| Организация работы | 0,02 | 0,3 | 30 000 | Принятие | Привлечение руководства | 20 000 |
|  |  |  |  |  |  | 150 000 |

## График проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 |  |  |  |  |
| 2 |  |  |  |  |
| 1 |  |  |  |  |
| 7 |  |  |  |  |
| 6 |  |  |  |  |
| 5 |  |  |  |  |
| 4 |  |  |  |  |
| 3 |  |  |  |  |
| 2 |  |  |  |  |
| 1 |  |  |  |  |
| 7 |  |  |  |  |
| 6 |  |  |  |  |
| 5 |  |  |  |  |
| 4 |  |  |  |  |
| 3 |  |  |  |  |
| 2 |  |  |  |  |
| 1 |  |  |  |  |
| 7 |  |  |  |  |
| 6 |  |  |  |  |
| 5 |  |  |  |  |
| 4 |  |  |  |  |
| 3 |  |  |  |  |
| 2 |  |  |  |  |
| 1 |  |  |  |  |
| 7 |  |  |  |  |
| 6 |  |  |  |  |
| 5 |  |  |  |  |
| 4 |  |  |  |  |
| 3 |  |  |  |  |
| 2 |  |  |  |  |
| 1 |  |  |  |  |
| Команда \ Дни | ПМ | Прораб | Архитектор | Бухгалтер |

## Бюджет проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Расходы на оплату труда** | **Должность** | **Рабочее время, ч** | **Ставка, р/ч** | **Стоимость** |
| Участник 1 | ПМ | 80 | 600 | 48 000 |
| Участник 2 | Прораб | 144 | 300 | 43 200 |
| Участник 3 | Архитектор | 56 | 500 | 28 000 |
| Участник 4 | Бухгалтер | 168 | 200 | 33 600 |
| **Итого** |  |  |  | **152 800** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Материальные расходы** |  |  |  |  |
| Материалы | Цемент, песок, кирпич |  |  | 3 500 000 |
| Субподрядчики | Бригада рабочих |  |  | 800 000 |
| Риски | Уход участника |  |  | 150 000 |
| Премии |  |  |  | 100 000 |
| **Итого** |  |  |  | **4 550 000** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Итого** |  |  |  | **4 702 800** |

# Унифицированный язык моделирования UML

UML (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

UML позволяет также разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий (таких как класс, компонент, обобщение (англ. generalization), агрегация (англ. aggregation) и поведение) и больше сконцентрироваться на проектировании и архитектуре.

Рассмотрим процесс UML проектирования на примере мультимедийной системы, осуществляющей хранение основной метаинформации для различных типов файлов (рисунок 4).

Построение системы начнем с абстракции *файла* – сущности, имеющей однозначное соответствие с реальным файлом на жестком диске. Класс MediaFile содержит в себе основные базовые характеристики файла: путь, размер, тип.

Современные медиасистемы зачастую распределенные, поэтому файл связан с определенным *сервером*. Сервер может быть в том числе и локальным, и предоставляет доступ до своих файлов по веб адресу.

Следующим логическим уровнем абстракции над файлами является *медиа объект*. Одному объекту Media могут быть сопоставлены несколько файлов (например, большое изображение и маленькое превью). У Media есть определенный тип, по которому можно получить дополнительную метаинформацию. Можно развивать систему дальше и добавить еще один группирующий слой. Например, несколько объектов Media можно объединить в рамках одного объекта Folder.

В системе поддерживаются 4 типа медиафайлов: видео, растровые и векторные изображения и аудио. Для каждого типа введем соответствующий ему класс, который будет инкапсулировать информацию, характерную только для этого конкретного типа.

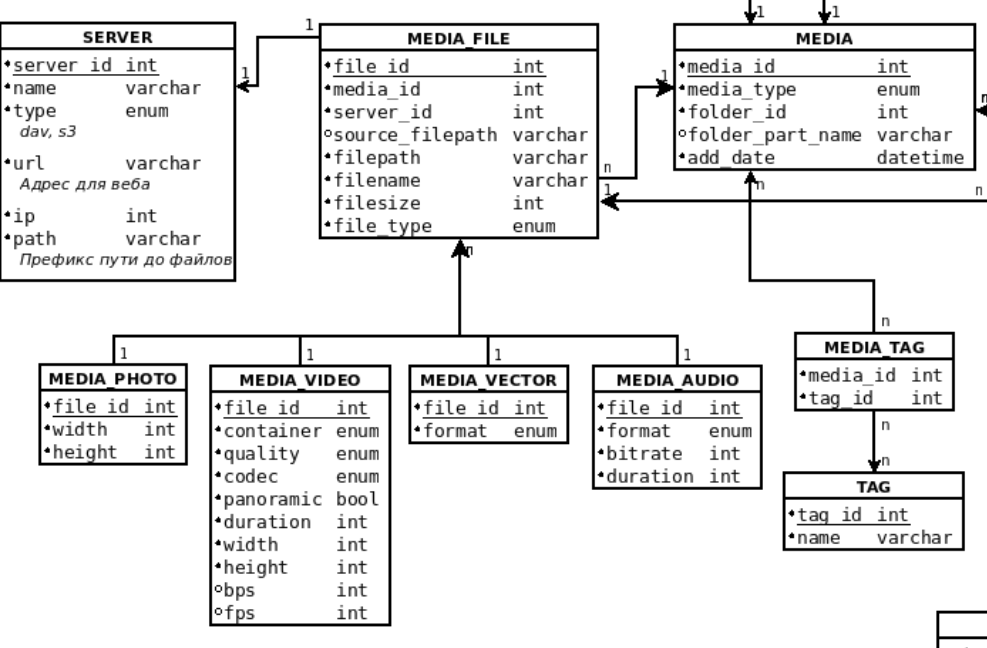


Рисунок 4 – UML диаграмма классов

Объектам Media можно назначать пользовательские *тэги* – простые текстовые описания, которые впоследствии могут использоваться при просмотре или поиске информации.

Даже на таком модельном примере видны основные преимущества UML. UML объектно-ориентирован, в результате чего методы описания результатов анализа и проектирования семантически близки к методам программирования на современных объектно-ориентированных языках. UML позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы. Диаграммы UML сравнительно просты для чтения после достаточно быстрого ознакомления с его синтаксисом. UML расширяет и позволяет вводить собственные текстовые и графические стереотипы, что способствует его применению не только в сфере программной инженерии.

UML получил широкое распространение и динамично развивается.

Список литературы

1. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования = Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development. — 3-е изд. — М.: Вильямс, 2006. — 736 с.
2. Джозеф Шмуллер. Освой самостоятельно UML 2 за 24 часа. Практическое руководство = Sams Teach Yourself UML in 24 Hours, Complete Starter Kit. — М.: Вильямс, 2005. — 416 с.
3. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. Язык UML. Руководство пользователя = The Unified Modeling Language user guide. — 2-е изд. — М., СПб.: ДМК Пресс, Питер, 2004. — 432 с.
4. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS / С. Орлов. — 2-е изд.. — СПб.: Питер, 2006. — 736 с.