**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники

Направление «Программная инженерия»

Отделение информационных технологий

Отчет по самостоятельной работе студента по дисциплине

**«Экономика программной инженерии»**

Выполнил:

Студент группы 8К61 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Исламов Е.Р.

Проверил:

Доцент ОИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ротарь В.Г.

Томск 2020

Содержание

[Содержание 2](#_Toc29941871)

[Самостоятельная работа студента 3](#_Toc29941872)

[Задание 1 – Построение графов с задачами на ребрах и вершинах 3](#_Toc29941873)

[Задание 2 – Разделение графа на слои. Алгоритм Демукрона 4](#_Toc29941874)

[Задание 3 – Удаление ошибок типа «КОНТУР». 5](#_Toc29941875)

[Задание 4 – Определение продолжительностей работ 6](#_Toc29941876)

[Задание 5 – Расчёт временных параметров событий 7](#_Toc29941877)

[Задание 6 – Расчет временных параметров работ 8](#_Toc29941878)

[Задание 7 – Управление ресурсами проекта 8](#_Toc29941879)

[Задание 8 – WBS-декомпозиция работ программного STARTUP-проекта 14](#_Toc29941880)

[Мудрости 17](#_Toc29941881)

[Мудрость №1 17](#_Toc29941882)

[Мудрость №2 24](#_Toc29941883)

[Мудрость №3 27](#_Toc29941884)

[Мудрость №4 29](#_Toc29941885)

[Мудрость №5 31](#_Toc29941886)

Самостоятельная работа студента

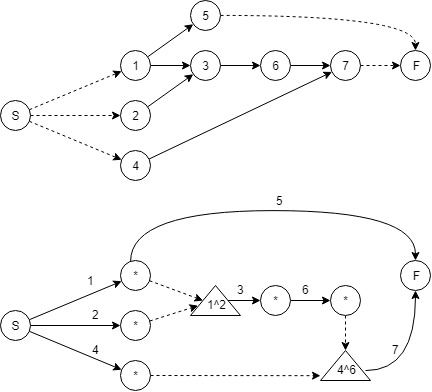
Задание 1 – Построение графов с задачами на ребрах и вершинах

На рис 1. представлена таблица, на основании которой производилось создание графов. Зеленым цветом выделены гласные буквы. В столбцах H и I отражены зависимости до и после оптимизации.



1. Генератор задач

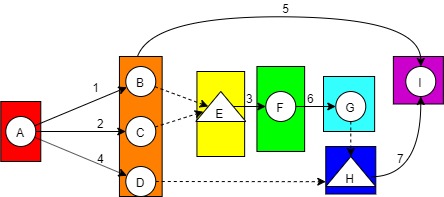
Графы с задачами на вершинах и на ребрах изображены на рис.2.



1. Графы с задачами на вершинах и на ребрах

Задание 2 – Разделение графа на слои. Алгоритм Демукрона

На рис.3 представлено ручное разбиение графа на слои.



1. Разбиение на слои

Воспользуемся алгоритмом Демукрона для разбиения графа на слои.

Таблица 1 – Таблица применения алгоритма Демукрона

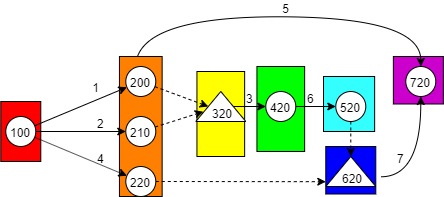
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | I |
| A | 1 | 1 | 1 | 1 |  |  |  |  |  |
| B | И | 1 |  |  | 1 |  |  |  | 1 |
| C | С |  | 1 |  | 1 |  |  |  |  |
| D | Т |  |  | 1 |  |  |  | 1 |  |
| E | О |  |  |  | 1 | 1 |  |  |  |
| F | К |  |  |  |  | 1 | 1 |  |  |
| G | И |  |  |  |  |  | 1 | 1 |  |
| H | С |  |  |  |  |  |  | 1 | 1 |
| I | Т | С | Т | О | К | С | Т | О | 1 |
| D0 | 1 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 3 |
| D1 |  | 1 | 1 | 1 | 3 | 2 | 2 | 3 | 3 |
| D2 |  |  |  |  | 2 | 2 | 2 | 3 | 2 |
| D3 |  |  |  |  | 1 | 2 | 2 | 3 | 2 |
| D4 |  |  |  |  |  | 2 | 2 | 2 | 2 |
| D5 |  |  |  |  |  | 1 | 2 | 2 | 2 |
| D6 |  |  |  |  |  |  | 1 | 2 | 2 |
| D7 |  |  |  |  |  |  |  | 1 | 2 |
| D8 |  |  |  |  |  |  |  |  | 1 |

Осуществим нумерацию вершин.

Таблица 2 – Нумерация вершин.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Вершины | Ранг | Номера |
| 1 | A | 0 | 100 |
| 2 | B, C, D | 1 | 200, 210, 220 |
| 3 | E | 2 | 320 |
| 4 | F | 3 | 420 |
| 5 | G | 4 | 520 |
| 6 | H | 5 | 620 |
| 7 | I | 6 | 720 |

На рисунке представлен Граф, получившийся в результате алгоритма Демукрона. Разделение на слои совпало с визуальным.



1. Разбиение на слои

Задание 3 – Удаление ошибок типа «КОНТУР».

Для диагноза наличия контуров в проекте воспользуемся итерационным алгоритмом. Смежность вершин отражена в таблице 3. Результат работы алгоритма представлен в таблице 4. Как следует из таблицы 4, контуры в графе отсутствуют.

Таблица 3 – Смежность вершин.

|  |  |
| --- | --- |
| **i** | **j** |
| A | B |
| A | C |
| A | D |
| B | E |
| B | I |
| C | E |
| D | H |
| E | F |
| F | G |
| G | H |
| H | I |

Таблица 4 – Итерационный алгоритм поиска циклов.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **k=0** | **k=1** | **k=2** | **k=3** | **k=4** | **k=5** | **k=6** | **k=7** |
| **A** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **B** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **C** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **D** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **E** | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| **F** | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| **G** | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| **H** | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| **I** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

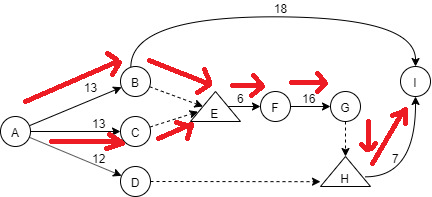
Задание 4 ­– Определение продолжительностей работ

Рассчитаем длительность работ. Результат расчета приведен в табл. 5.

Таблица 5 – Длительности работ

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| a | И/10 | В/3 | А/1 | В/3 | Е/6 | И/10 | В/3 |
| m | Л/13 | М/14 | Е/6 | Л/13 | С/19 | О/16 | Е/6 |
| b | О/16 | С/19 | И/10 | Н/15 | Ч/25 | У/21 | М/14 |
| t(в днях) | 13 | 13 | 6 | 12 | 18 | 16 | 7 |

Обозначим продолжительности работ на графе и выделим критический путь (рис. 9).



1. Определение критического пути

Было найдено 2 критических пути:

A -> B -> E -> F -> G -> H -> I

A -> C -> E -> F -> G -> H -> I

Длина совпадает и равна: 13 + 6 + 16 + 7 = 42

Задание 5 – Расчёт временных параметров событий

Произведем расчет раннего, позднего срока свершения событий, а также расчет резерва времени событий.

Таблица 5 – Сроки свершения и резервы.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | I | tp 0 | tp 1 | tp 2 |
| A |  | 13 | 13 | 12 |  |  |  |  |  | 0 | 0 | 0 |
| B |  |  |  |  | 0 |  |  |  | 18 | 0 | 13 | 13 |
| C |  |  |  |  | 0 |  |  |  |  | 0 | 13 | 13 |
| D |  |  |  |  |  |  |  | 0 |  | 0 | 12 | 12 |
| E |  |  |  |  |  | 6 |  |  |  | 0 | 13 | 13 |
| F |  |  |  |  |  |  | 16 |  |  | 0 | 19 | 19 |
| G |  |  |  |  |  |  |  | 0 |  | 0 | 35 | 35 |
| H |  |  |  |  |  |  |  |  | 7 | 0 | 35 | 35 |
| I |  |  |  |  |  |  |  |  |  | 0 | 42 | 42 |
| tn 0 | 42 | 42 | 42 | 42 | 42 | 42 | 42 | 42 | 42 |  |  |  |
| tn 1 | 0 | 13 | 13 | 35 | 13 | 19 | 35 | 35 | 42 |  |  |  |
| tn 2 | 0 | 13 | 13 | 35 | 13 | 19 | 35 | 35 | 42 |  |  |  |
| P(i) | 0 | 0 | 0 | 23 | 0 | 0 | 0 | 0 | 0 |  |  |  |

В столбце tp 2 можно увидеть срок раннего свершения событий, в строке tn 2 – срок позднего свершения событий, а в строке P(i) – резерв времени.

Задание 6 – Расчет временных параметров работ

Проведем расчет сроков начала и окончания работы. В столбцах tрн, tро, tпн, tпо отражены сроки раннего начала, раннего окончания, позднего начала и позднего окончания соответственно.

Проведем расчет резервов времени задач. В столбцах Рп, Рп’, Рп’’, Рс приведены полный резерв времени, частный резерв времени первого рода, резерв второго рода и свободный резерв соответственно.

Проведем расчет коэффициентов резервов времени. В столбцах Кп, Кп’, Кп’’, Кс, Кн приведены коэффициент полного резерва времени, коэффициент частного резерва времени первого, коэффициент второго порядка, коэффициент свободного резерва и коэффициент напряженности.

Таблица 6 – Временные параметры работ

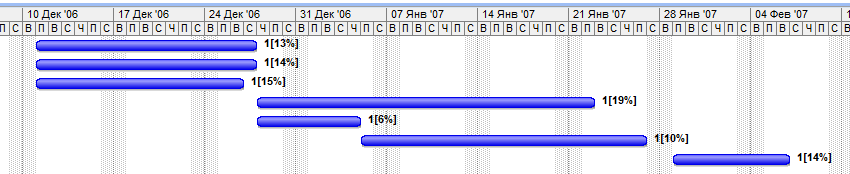


Задание 7 – Управление ресурсами проекта

Сформируем интенсивности работ из второго столбца матрицы генератора СРС1.

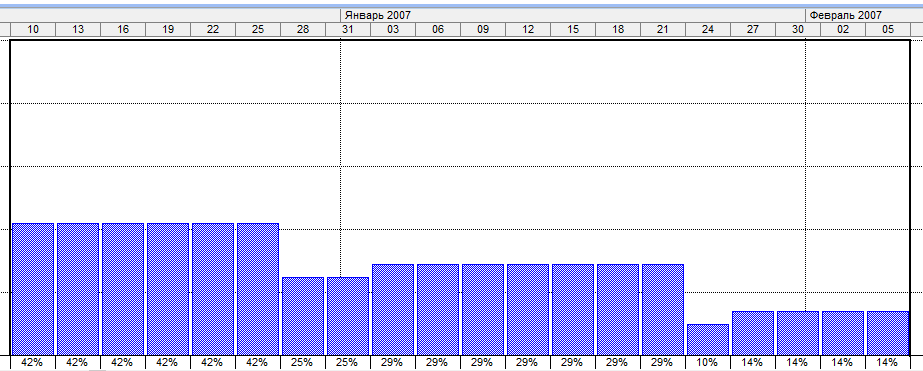
|  |  |  |
| --- | --- | --- |
| **Дуга** | **Буква** | **V** |
| A-B | Л | 13 |
| A-C | М | 14 |
| E-F | Е | 6 |
| A-D | Н | 15 |
| B-I | С | 19 |
| F-G | И | 10 |
| H-I | М | 14 |

Диаграмма Ганта при условии неограниченных ресурсов представлен на рисунке 13.



1. Диаграмма Ганта

Построим график потребления ресурсов использованием пакета MS Project (рис. 7).



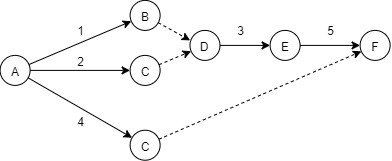
1. График потребления ресурсов

Составим сетевую модель объединённого проекта по генератору (рис. 16). Столбцы t и v длительности и интенсивности формировались на основании столбцов 2 и 3.



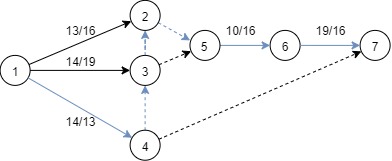
1. Генератор объединённого проекта

На рисунке 9 представлено графическое изображение сетевой модели объединенного проекта.



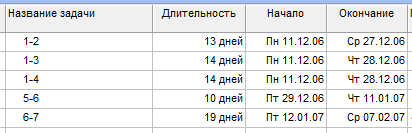
1. Объединенный проект

Вводим добавочные связи для того, чтобы появилась цепь, проходящая через все события (рис. 10).



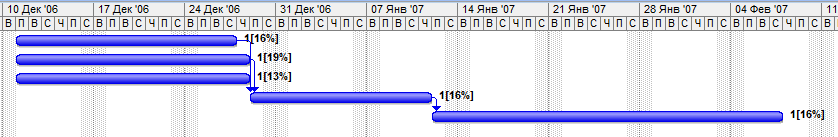
1. Дополненный объединенный проект.

Для моделирования графа используется пакет MS Project. Зададим задачи (рис. 11).

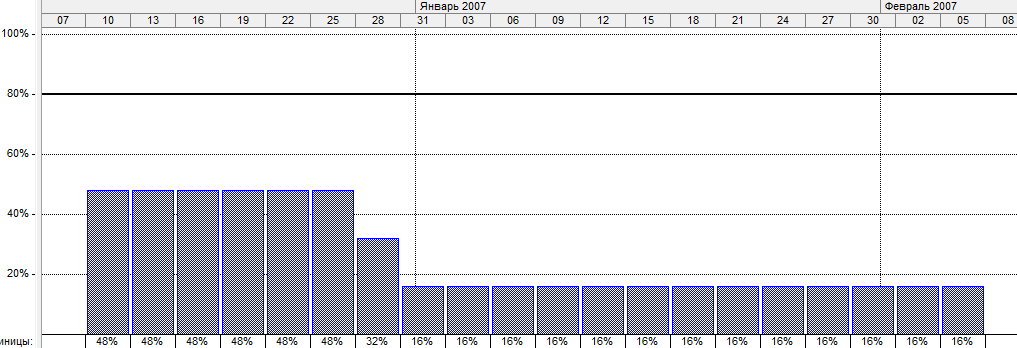


1. Список задач в MS Project

Построим диаграмму Ганта и график потребления ресурсов (рис. 12, 13).



1. Диаграмма Ганта до наложения ограничений.



1. График использования ресурсов до наложения ограничений.

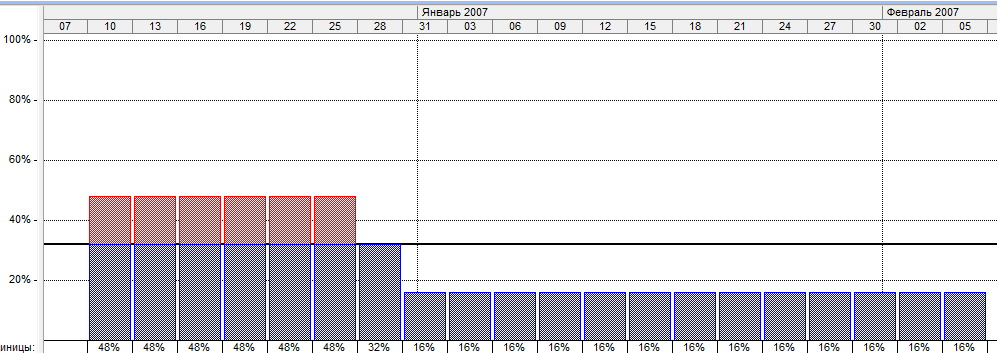
Длительность составляет 43 рабочих дня.

Составим таблицу фронтов работ.

Таблица 7 – фронты работ

|  |  |  |
| --- | --- | --- |
| k | Фк | vk |
| 1 | 1,2 | 16 |
| 2 | 1,3 | 19 |
| 3 | 1,4 | 13 |
| 4 | 5,6 | 16 |
| 5 | 6,7 | 16 |
| 6 | 1,2; 1,3 | 35 |
| 7 | 1,2; 1,4 | 29 |
| 8 | 1,3; 1,4 | 32 |
| 9 | 1,2; 1,3; 1,4 | 48 |

Как следует из графика, Rmax = 48. Максимальная интенсивность атомарной задачи равна 33. Значит, предельная интенсивность потребления A = max(48/2, 19) = 24. Однако, вопреки предлагаемой формуле, установим предельную интенсивность 32, т.к. при интенсивности 24 работы не смогут выполняться параллельно. A = 32.



1. График использования ресурсов после наложения ограничений.

Фронты 6, 9 недопустимы по интенсивности. Значит, длительность этих фронтов равна нулю. Решим задачу нахождения минимума функции симплекс-методом.

Для решения задачи воспользуемся языком программирования R. Исходный код программы приведен ниже:

install.packages("lpSolve")

library(lpSolve)

f.obj <- c(1, 1, 1, 1, 1, 1, 1)

f.con <- matrix(c(1, 0, 0, 0, 0, 1, 0,

0, 1, 0, 0, 0, 0, 1,

0, 0, 1, 0, 0, 1, 1,

0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 1, 0, 0), nrow=5, byrow=TRUE)

f.dir <- c("=",

"=",

"=",

"=",

"=")

f.rhs <- c(13, 14, 14, 10, 19)

result <- lp ("min", f.obj, f.con, f.dir, f.rhs)

print(result)

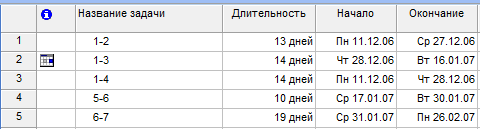
print(result$solution)

Дополним таблицу 7 результатами работы программы.

Таблица 8 – фронты работ и коэффициенты при минимальном значении результирующей функции

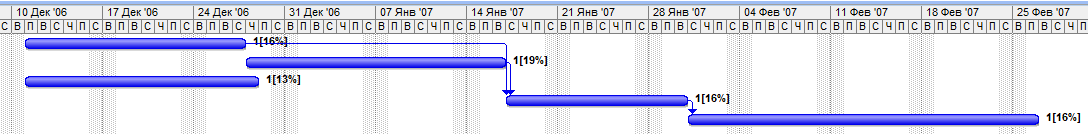
|  |  |  |  |
| --- | --- | --- | --- |
| k | Фк | vk | X |
| 1 | 1,2 | 16 | 0 |
| 2 | 1,3 | 19 | 13 |
| 3 | 1,4 | 13 | 0 |
| 4 | 5,6 | 16 | 10 |
| 5 | 6,7 | 16 | 19 |
| 6 | 1,2; 1,3 | 35 | 0 |
| 7 | 1,2; 1,4 | 29 | 13 |
| 8 | 1,3; 1,4 | 32 | 1 |
| 9 | 1,2; 1,3; 1,4 | 48 | 0 |

На рисунках 15, 16, 17 представлены список задач, диаграмма Ганта и график использования ресурсов, получившиеся после адаптации к ограничениям с использованием симплекс-метода.

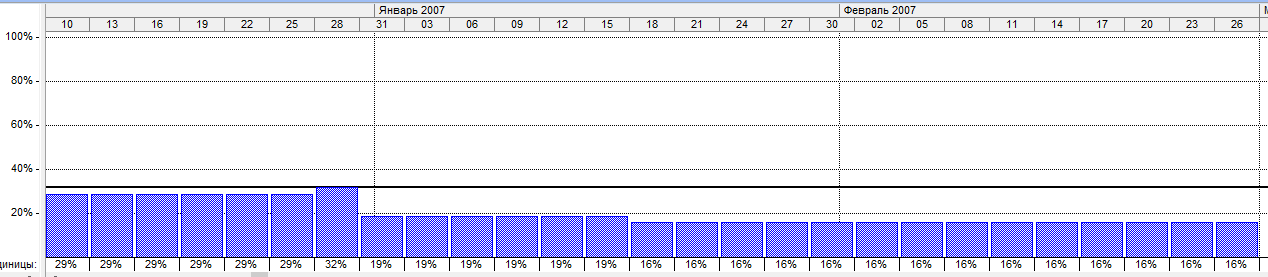


1. Список задач после адаптации к ограничениям

Построим диаграмму Ганта и график потребления ресурсов (рис. 12, 13).



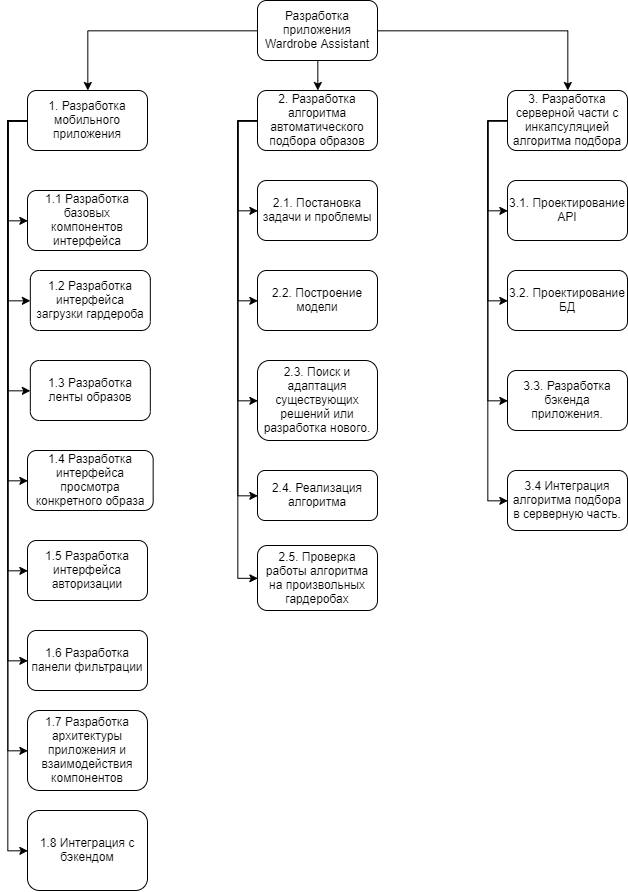
1. Диаграмма Ганта после адаптации к ограничениям



1. График использования ресурсов после адаптации к ограничениям

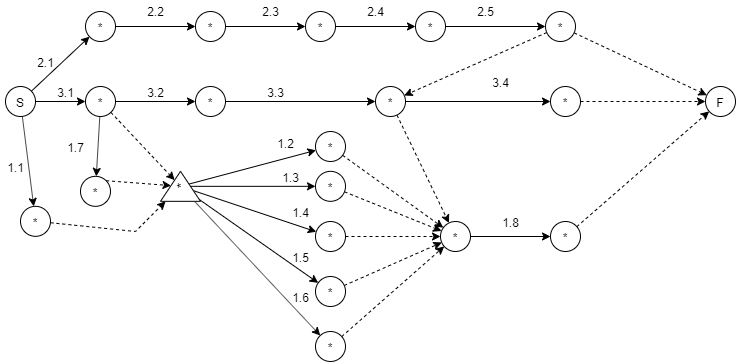
Задание 8 – WBS-декомпозиция работ программного STARTUP-проекта

STARTUP-проектом подгруппы является мобильное приложение для автоматического подбора образов из имеющегося гардероба Wardrobe Assistant. На рисунке 18 представлен результат декомпозиции.



1. WBS проекта

На рисунке 19 представлена сетевая модель с задачами на ребрах.



1. Сетевая модель проекта

Мудрости

Мудрость №1

«Измерение размера программного обеспечения»

**Измерение размера ПО**

Среди факторов, влияющих на оценку, размер проекта является наиболее важным показателем. Хотя оценки размера недостаточно для понимания общей сложности разрабатываемого продукта, существует явная зависимость между размером проекта и его трудоемкостью.

Чтобы определить, по каким показателям можно сравнивать метрики размера ПО, необходимо рассмотреть основные группы метрик подробнее.

**Количество строк кода**

Количество строк кода – это размерно-ориентированная метрика программного обеспечения, в которой объем ПО рассчитывается исходя из количества строк в тексте исходного кода.

Эта методика возникла в 1950-е годы. Основным носителем информации в те времена была перфокарта, причем на одной перфокарте кодировалась одна строка исходного кода. Поскольку каждая строка кода являлась отдельным физическим носителем, можно было подсчитать число этих объектов и определить трудоемкость и производительность труда программистов.

Среди методик учета числа строк есть две основные:

* по числу физических строк (LOC) – определяется как общее число строк исходного кода, включая комментарии и пустые строки;
* по числу логических строк кода (LLOC) – определяется как общее количество команд и зависит от используемого языка программирования. Если язык поддерживает размещение нескольких команд в одной строке, то одна физическая строка должна быть учтена как несколько логических, если она содержит более одной команды языка.

Также имеются производные от основных методик, которые в зависимости от задачи могут содержать дополнительную информацию по следующим показателям:

* число пустых строк;
* число строк, содержащих комментарии;
* процент комментариев (отношение строк кода к строкам комментария, производная метрика стилистики);
* среднее число строк для функций (классов, файлов);
* среднее число строк, содержащих исходный код для функций (классов, файлов);
* среднее число строк для модулей.

У оценки по количеству строк кода есть несколько минусов. В современных высокоуровневых языках одну и ту же функциональность можно описать различным числом строк кода, поэтому для них данная метрика может слабо коррелировать с реальными трудозатратами.

Кроме того, при использовании современных средств разработки ПО часто используется генерация кода для определенных действий, что может еще сильнее усложнить итоговую взаимосвязь между числом строк и трудоемкостью.

При этом у измерений в строках кода есть ряд преимуществ. Например, данные по количеству строк в завершенных проектах или модулях программ могут быть легко собраны при помощи служебных средств интегрированных сред разработки (IDE) или специальных программ.

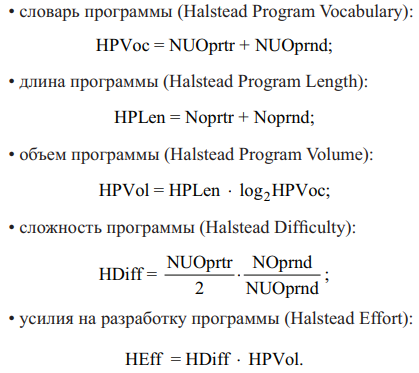
**Метод Холстеда**

Метрики, основанные на анализе числа строк и синтаксических элементов исходного кода программы, были предложены М. Холстедом (MauriceHalstead) в 1977 г.. Метрики Холстеда (Halstead complexity measures) частично позволяют учесть возможность реализации одной и той же функциональности разным числом строк и операторов. Наиболее частым сценарием использования этих метрик является оценка сложности промежуточных продуктов разработки, однако набор метрик также содержит оценки размера.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

* NUOprtr (Number of Unique Operators) – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);
* NUOprnd (Number of Unique Operands) – число уникальных операндов программы (словарь операндов);
* Noprtr (Number of Operators) – общее число операторов в программе;
* Noprnd (Number of Operands) – общее число операндов в программе.

На основе этих характеристик вычисляют различные метрики размера, сложности и качества программ, такие как:



Хотя метрики Холстеда позволяют использовать дополнительные возможности по анализу исходного кода, которые отсутствуют в метрике числа строк кода, с точки зрения оценки размера проекта большая часть задач остается нерешенной. Оценка общего числа операторов и их операндов до завершения этапа разработки программы может оказаться еще более сложной задачей, чем оценка числа строк кода.

**Функциональные точки**

Анализ функциональных точек (Function points) – это метод измерения размера программного обеспечения с точки зрения пользователей системы 8. Метод был разработан Аланом Альбрехтом (Alan Albrecht) в середине 1970-х годов, впервые опубликован в 1979 г. Широкое распространение эта методика получила в середине 1980-х годов, после того как была сформирована организация IFPUG, занимающаяся развитием метода, а также публикующая новые версии 9.

Данный метод предназначен для оценки объема программного продукта по функциональной модели, т. е. оценивается объем функций разрабатываемой системы. Основная цель этого метода – в декомпозиции системы таким образом, чтобы обеспечить приемлемую сложность анализа. Базовой единицей измерения, на которой основывается данный метод, является функциональная точка. Оценка размера в функциональных точках базируется на количестве и сложности следующих элементов:

• внешних входных элементов – всех элементов, предназначенных для ввода или управления данными, которые поступают в систему;

• внешних выходных элементов – всех элементов для ввода и управления данными, которые выходят за внешние границы системы;

• внешних запросов – комбинации входных и выходных элементов, в которых входной элемент сопоставляется с простой выходной формой;

• внутренних логических файлов – каждого логического файла (группы данных), который создается или используется в системе; •

• внешних интерфейсных файлов – каждого файла под управлением другой системы, с которым взаимодействует измеряемая программа

Анализ функциональных точек включает в себя следующие этапы 10:

1) определения типа оценки;

2) определения области оценки и границ продукта;

3) подсчета функциональных точек, связанных с данными;

4) подсчета функциональных точек, связанных с транзакциями;

5) определения суммарного количества не выровненных функциональных точек (UFP);

6) определения значения коэффициента выравнивания (FAV);

7) расчета количества выровненных функциональных точек (AFP).

На двух первых этапах анализируется предмет оценки, определяются границы продукта, выявляется разрабатываемая функциональность.

На этапах 3–5 происходят общий подсчет и суммирование функциональных точек без учета коэффициента выравнивания.

При выполнении 6-го пункта в методе расчета вводятся общесистемные требования, которые накладывают различные ограничения и увеличивают сложность разработки.

Сложность этих требований оцениваются коэффициентом выравнивания (FAV), который зависит от 14 общих системных характеристик (total degree of infl uence, TDI) и вычисляется по формуле



В целом измерение размера проекта функциональными точками является более актуальной метрикой, чем измерение числом строк кода. Ключевое преимущество этого подхода – оценка основана на требованиях к продукту, что позволяет оценить трудоемкость на самых ранних этапах работы над проектом, сразу после выявления необходимых требований. На последующих этапах работы оценку можно уточнить, поэтому этот метод можно применять при использовании гибких методологий разработки ПО. Также надо отметить, что декомпозиция системы, выполняемая на начальных этапах расчета, может в дальнейшем стать основой для документирования функциональности системы, а значит, снизить трудозатраты проектной команды. Однако использование функциональных точек в качестве единиц измерения имеет ряд существенных недостатков. Для вычисления функциональных точек необходимо детально изучить спецификацию требований и подсчитать все входные и выходные элементы, файлы, транзакции, что может быть весьма трудоемко. При этом полученная оценка весьма субъективна, поскольку некоторые бизнес-процессы могут иметь высокую алгоритмическую сложность, но обладать достаточно простыми внешними вводами и выводами. По мере развития методологий разработки программного обеспечения эксперты предложили новые методы оценки, основанные на методе функциональных точек.

**Объектные точки**

В современных методологиях разработки объектно-ориентированное программирование (ООП) занимает особое место, фактически является наиболее широко распространенной методологией.

Поскольку в изначальном варианте метода функциональных точек не было предусмотрено применения объектно-ориентированного подхода, был разработан адаптированный вариант, оперирующий терминами ООП.

Его принципиальным отличием от других вариаций метода функциональных точек является то, что он не расширяет стандартный набор типов элементов, а использует совершенно иные 13:

• формы (Screens definitions);

• отчеты (User reports);

• модули (3GL Modules).

Фактически в данной метрике каждому уникальному классу или объекту назначается одна объектная точка. В целом оценка производится примерно по тем же этапам, что и функциональные точки: 1) подсчет количества форм, отчетов и компонентов; 2) классификация каждого экземпляра объекта по уровню сложности; 3) определение веса для объектов; 4) суммирование взвешенных объектов; 5) определение процента повторного использования кода; 6) определение уровня продуктивности; 7) подсчет значения длительности работы в человекомесяцах.

По сравнению с методом функциональных точек ощутимые различия есть при использовании факторов преобразования на этапах 5–7. Во-первых, подсчитывается процент повторного использования, исходя из него количество объектных точек пересчитывается

Этапы:

1) Подсчет количества форм, отчетов и компонентов;

2) Классификация каждого экземпляра объекта по уровню

сложности;

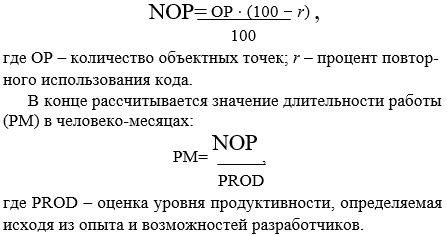
3) Определение веса для объектов;

4) Суммирование взвешенных объектов;

5) Определение процента повторного использования кода;

6) Определение уровня продуктивности;

7) Подсчет значения длительности работы в человекомесяцах.



**Метод UCP**

Метод UCP (Use-case points) представляет собой оценку размера проектов на основе диаграмм UML (Unified Modeling Language) и методологии RUP (Rational Unified Process). Как и многие другие современные методы оценки, UCP базируется примерно на тех же принципах, что и метод функциональных точек. Главное различие заключается в замене единиц измерения с функциональных точек на варианты использования (Use Cases).

Оценка по методу UCP складывается из следующих эта-пов 14:

1. Оценка акторов. На этом шаге определяются все акторы системы (сущности, взаимодействующие с системой извне). После определения для каждого актора в соответствии с его оценкой устанавливается вес.
2. Нескорректированная оценка вариантов использования. Рассчитывается исходя из количества транзакций.
3. Оценка технических факторов. Используется для определения сложности архитектуры приложения и степени влияния нефункциональных требований.
4. Оценка внешних факторов. Используется для опреде-ления коэффициента влияния организационных рисков на разработку.
5. Окончательный подсчет. Оценивается общее число вариантов с учетом прочих факторов по формуле



где UUCW – нескорректированная оценка вариантов исполь-зования; UAW – оценка акторов; TCF – оценка технических факторов; ECF – оценка внешних факторов.

Мудрость №2

«Принципы алгоритмического моделирования трудоемкости разработки программных продуктов»

**Теоретические модели**

Математическое моделирование трудоемкости разработки ПО основано на сопоставлении экспериментальных данных с формой существующей математической функции. В начале 1960-х годов Питер Норден из фирмы IBM пришел к выводу, что в проектах по исследованию и разработке может применяться хорошо прогнозируемое распределение трудовых ресурсов, основанное на распределении вероятности, называемом кривой Рэлея (Rayleigh distribution). Позднее, в 1970-х годах Лоуренс Патнэм1 из компании Quantitative Systems Management применил результаты Нордена к разработке ПО. Используя статистический анализ проектов, Патнэм обнаружил, что взаимосвязь между тремя основными параметрами проекта (размером, временем и трудоемкостью) напоминает функцию Нордена-Рэлея (рис. 1), отражающую распределение трудовых ресурсов проекта в зависимости от времени.

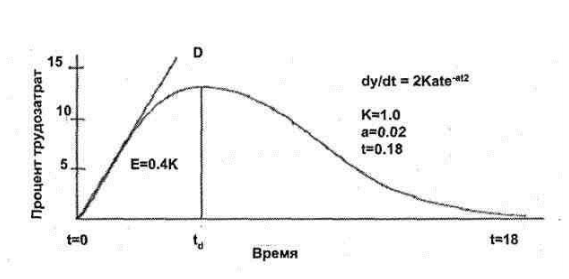
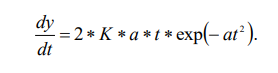


Рисунок 1 - функция Рэлея

Функция Рэлея моделируется дифференциальным уравнением:

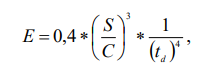


где dy/dt — скорость роста персонала проекта; t — время, прошедшее от начала проекта до изъятия продукта из эксплуатации; К — область под кривой — представляет полную трудоемкость в течение всего жизненного цикла (включая сопровождение), выраженную в человеко-годах; а — константа, которая определяет форму кривой (фактор ускорения) и вычисляется по формуле:



Где t - время разработки.

Приняв ряд допущений, Патнэм получил следующее уравнение:



где Е — трудоемкость разработки ПО, S — размер ПО в строках кода, td— планируемый срок разработки, С — технологический фактор, учитывающий различные аппаратные ограничения, опыт персонала и характеристики среды программирования. Он определяется на основе хронологических данных по прошлым проектам и, согласно рекомендациям Патнэма определяется для различных 501 типов проектов следующим образом:

• проект, внедренный в сжатые сроки без детальной проработки, -1500;

• проект, выполненный в соответствии с четким планом, — 5000;

• проект, предусматривающий оптимальную организацию и поддержку, — 10000. Оптимальный срок разработки определяется как



что хорошо согласуется с большинством статистических моделей.

Более подробное описание модели Патнэма приведено в книге Фатрелл Р., Шафер Д., Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат: Пер. с англ. — М.: Вильяме, 2003.

**Статистические модели**

Статистические модели используют накопленные хронологические данные, чтобы получить значения для коэффициентов модели. Для определения соотношений между параметрами модели и трудоемкостью разработки ПО используется регрессионный анализ. Существуют две формы статистических моделей: линейные и нелинейные.

Линейные статистические модели имеют следующий вид:



где xi — факторы, влияющие на трудоемкость, bi— коэффициенты модели.

Линейные модели работают не слишком хорошо, поскольку практика показывает, что соотношения между трудоемкостью и размером ПО нелинейно. По мере роста размера ПО возникает экспоненциальный отрицательный эффект масштаба.

Нелинейные, статистические модели имеют следующий вид:



где А — комбинация факторов, влияющих на трудоемкость; b — экспоненциальный коэффициент масштаба.

Статистические модели просты для понимания, но имеют следующий недостаток: результаты справедливы в основном только для конкретной ситуации. Другой недостаток — при увеличении количества входных параметров количество данных, необходимых для калибровки модели, также возрастает.

Примером нелинейной модели является модель COCOMO. На базовом уровне трудоемкость рассчитывается по формуле:

**Трудоемкость** = ab(KLOC)bb **[человеко-месяцев]**

На среднем уровне трудоемкость рассчитывается по формуле:

**Трудоемкость =*ai*(KLoC)*(bi)*\*РФТ**

Мудрость №3

«Риски проведения оценки разработки программного обеспечения»

На основе изучения «ГОСТ Р ИСО/МЭК 12207-99 Информационная технология. Процессы жизненного цикла программных средств» были выявлены основные процессы разработки программного обеспечения:

* выявление и анализ требований;
* проектирование программного обеспечения;
* программирование;
* тестирование программного обеспечения.

Оценка проекта производится после стадии выявления и анализа требований, поэтому ошибки при анализе требований влияют на оценку проекта. Остальные 3 пункта не связаны с оценкой проекта, а совершаются после нее.

**Выявление и анализ требований**

Выявление и анализ требований играют важную роль в успехе проекта разработки программного обеспечения, так как большая часть ошибок происходит на ранних стадиях разработки.  
Это связано с природой разработки программного обеспечения: выполняемая работа базируется на ранее выполненной. Например, проектирование выполняется на основе требований, программирование реализуется на основе модели проектирования, а тестирование производится на основе написанного текста программы.

1. *Непонимание концепции проекта*

Если лица, заинтересованные в проекте не имеют ясного мнения о том, что должен представлять собой продукт, то вероятность расползания границ системы возрастает

*2. Двусмысленные требования*

Первый вариант двусмысленности заключается в том, что пользователь может интерпретировать одно и то же требование по-разному. Второй вариант состоит в том, что у нескольких читателей возникает разное понимание того, что означает требование.

*3. Изменение требований.*

Изменение понимания пользователями и разработчиками решаемых системой задач, изменения в деловой среде, техническом, программном обеспечении

*4. Недостаточное уделение внимания нефункциональным требованиям*

Основное внимание уделяется на функциональность продукта

**Риски при оценке ПО**

При оценке ПО могут быть следующие риски, способные повлиять на успех продукта:

1. В задачу не заложена обработка ошибок
2. Не учтены краевые состояния продукта
3. Проект недостаточно разбит на этапы и задачи внутри каждого этапа. Это может привести к «грубой» оценке, которая не будет соответствовать действительности. Для точной оценки необходимо проводить тщате.
4. В оценку не закладываются часы на проектную коммуникацию команды. Обычно она занимает 15-20% от общего срока.
5. Добавить часы на стабилизацию после завершения каждого этапа. Стабилизация — не риск, а типичная задача проекта. Это ещё 20-25% от времени, запланированного на реализацию.
6. Не заложен буфер на непредвиденные обстоятельства. Если проект ведётся по модели Fixed Price, сверху необходимо заложить дополнительный буфер в 20-30% на непредвиденные обстоятельства.
7. Недостаточная верификация оценки продукта. Правдоподобная оценка — та, которую трижды верифицировали: команда разработки, менеджер и руководитель.

Мудрость №4

«Зрелость процессов разработки программного обеспечения в системе CMMI»

Capability Maturity Model Integration (CMMI) — набор моделей (методологий) совершенствования процессов в организациях разных размеров и видов деятельности. CMMI содержит набор рекомендаций в виде практик, реализация которых, по мнению разработчиков модели, позволяет реализовать цели, необходимые для полной реализации определённых областей деятельности.

Набор моделей CMMI включает три модели: CMMI for Development (CMMI-DEV), CMMI for Services (CMMI-SVC) и CMMI for Acquisition (CMMI-ACQ). Наиболее известной является модель CMMI for Development, ориентированная на организации, занимающиеся разработкой программного обеспечения, аппаратного обеспечения, а также комплексных систем. На данный момент действуют две версии моделей: три отдельные модели версии 1.3 (вышли в ноябре 2010 года) и частично опубликованная единая версия 2.0 (вышла в конце марта 2018 года).

**Структура**

CMMI определяет 22 процессные области (process areas). Для каждой из процессных областей существует ряд целей (goals), которые должны быть достигнуты при внедрении CMMI в данной процессной области. Некоторые цели являются уникальными — они называются специфическими (specific). Общие (generic) цели применяются ко всем процессным областям. Цели достигаются при помощи реализации практик (рекомендаций модели), либо их адекватных альтернатив. В соответствии с отношением к какой-либо цели, практики делятся на специфические и общие.

Существуют два представления CMMI: непрерывное (continuous) и ступенчатое (staged). При реализации практик CMMI с использованием непрерывного представления, выбор процессных областей не фиксирован (хотя в модели содержатся рекомендации по порядку реализации областей). Для оценки уровня институционализации процессной области используется шкала уровней способности (capability level) от 0 до 5 (шесть уровней). Ступенчатое представление определяет пять (1-5) уровней зрелости (maturity level) организации. Для достижения каждого уровня зрелости (кроме первого) необходимо выполнить требования по реализации целей определённого набора процессных областей для достижения соответствующих целей. Первый уровень зрелости в модели не определён.

**Система измерения зрелости**

Любое совершенствование процессов подразумевает плавный/поэтапный процесс. В CMMI эти этапы формализованы — существует 5 уровней зрелости, каждый из которых указывает на зрелость процессов организации.

|  |  |
| --- | --- |
| 5. Фокус на совершенствование процессов | Оптимизируемый |
| 4. Процессы измеряются и контролируются | Управляемый на основе количественных данных |
| 3. Процессы определены на уровне всей организации. Процессы исполняются заблаговременно | Определенный |
| 2. Процессы определены на уровне проектов. Зачастую процессы появляются в ответ на определенные события | Управляемый |
| 1. Процессы непредсказуемые, слабо контролируемые. процессы появляются в ответ на определенные события | Начальный |

Мудрость №5

«Особенности управления проведением экспертных оценок»

При использовании метода экспертных оценок возникают свои проблемы. Основными из них являются: подбор экспертов, проведение опроса экспертов, обработка результатов опроса, организация процедур экспертизы.

Каждая из этих проблем, в свою очередь, содержит совокупность подпроблем.

Первым этапом организации работ по применению экспертного оценивания является подготовка и издание руководящего документа, в котором формулируется цель работы и основные положения по ее выполнению. В этом документе должны быть отражены следующие вопросы: постановка задачи эксперимента; цели эксперимента; обоснование необходимости эксперимента; сроки выполнения работ; задачи и состав группы управления; обязанности и права группы; финансовое и материальное обеспечение работ.

Для подготовки этого документа, а также для руководства всей работой назначается руководитель экспертизы. На него возлагается формирование группы управления и ответственность за организацию ее работы.

После формирования группа управления осуществляет работу по подбору экспертной группы примерно в такой последовательности: уяснение решаемой проблемы; определение круга областей деятельности, связанных с проблемой; определение долевого состава экспертов по каждой области деятельности; определение количества экспертов в группе; составление предварительного списка экспертов с учетом их местонахождения; анализ качеств экспертов и уточнение списка экспертов в группе; получение согласия экспертов на участие в работе; составление окончательного списка экспертной группы.

Параллельно с процессом формирования группы экспертов группа управления проводит разработку организации и методики проведения опроса экспертов. При этом решаются следующие вопросы: место и время проведения опроса; количество и задачи туров опроса; форма проведения опроса; порядок фиксации и сбора результатов опроса; состав необходимых документов.

Следующим этапом работы группы управления является определение организации и методики обработки данных опроса. На данном этапе необходимо определить задачи и сроки обработки, процедуры и алгоритмы обработки, силы и средства для проведения обработки.

В процессе непосредственного проведения опроса экспертов и обработки его результатов группа управления осуществляет выполнение комплекса работ в соответствии с разработанным планом, корректируя его по мере необходимости по содержанию, срокам и обеспечению ресурсами.

Последним этапом работ для группы управления является оформление результатов работы. На этом этапе производится анализ результатов экспертного оценивания; составление отчета; обсуждение и одобрение ре­зультатов; представление итогов работы на утверждение; ознакомление с результатами экспертизы организа­ций и лиц.