Д. Н. Ковалев

ПРИМЕНЕНИЕ СЕТЕВОГО ПЛАНИРОВАНИЯ ДЛЯ УПРАВЛЕНИЯ РАЗРАБОТКОЙ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

Рассматривается применение сетевого планирования к проекту разработки мобильного приложения для Android и iOS. Построена сетевая модель, включающая этапы анализа требований, UX/UI-проектирования, серверной и клиентской разработки, тестирования и публикации. Выполнен расчёт ранних и поздних сроков начала работ, определены резервы времени и критический путь.

Введение

Разработка мобильного приложения для Android и iOS является сложным ИТ-проектом, включающим аналитические, проектные, программные, тестовые и организационные работы. Такие проекты редко выполняются линейно: параллельно могут разрабатываться интерфейс, серверная часть, API, графические материалы, облачная инфраструктура и документация.

При отсутствии формализованного календарного плана задержка одной работы может привести к переносу срока запуска всего продукта. Поэтому актуальной является задача применения методов сетевого планирования, позволяющих определить минимальную длительность проекта, критический путь и резервы времени некритических работ [1].

Цель работы – построить и проанализировать сетевую модель проекта разработки мобильного приложения, а также показать, как результаты расчёта могут использоваться для распределения работ между основной командой и внешними исполнителями.

I. Постановка задачи

В качестве объекта исследования выбран проект разработки мобильного приложения для Android и iOS. В сетевую модель включены 16 работ: инициация проекта; анализ рынка и конкурентов; разработка технического задания; разработка UX/UI-прототипа; проектирование архитектуры базы данных; разработка серверной части; разработка API; разработка клиентской части мобильного приложения; подготовка графического контента; настройка облачной инфраструктуры; функциональное тестирование; аудит безопасности API; устранение выявленных дефектов; подготовка юридической документации; публикация приложения в App Store и Google Play; официальный запуск. Нумерация вершин на сетевом графике соответствует порядку перечисления работ.

Выбранный перечень работ отражает реальный процесс создания программного продукта [2]. Серверная часть и пользовательский интерфейс могут проектироваться параллельно, однако клиентская часть зависит от готовности API. Тестирование возможно только после завершения разработки мобильного клиента и подготовки графики, а публикация приложения требует исправления дефектов и готовности юридической документации [3].

II. Методика расчёта

Сетевое планирование основано на представлении проекта в виде ориентированного графа, где вершины соответствуют работам, а дуги отражают зависимости между ними. Для каждой работы задаётся длительность и перечень непосредственно связанных работ.

Обозначим t_i время выполнения работы i, $T(i)$ – раннее время начала работы i, $T(i)$ – позднее время начала работы i. Раннее время начала определяется прямым проходом по сети:

$$T(i) = \max_{j \in G} \left(T(j) + t_j \right), \tag{1}$$

где G – множество работ, непосредственно предшествующих работе i.

Позднее время начала определяется обратным проходом:

$$T(i) = \min_{j \in H} \left(T(j) - t_i \right), \tag{2}$$

где H – множество работ, непосредственно следующих за работой i.

Резерв времени рассчитывается как

$$R(i) = T(i) - T(i). \tag{3}$$

Работы с нулевым резервом образуют критический путь, определяющий минимальную длительность проекта.

III. Построение сетевой модели

Построенный сетевой график проекта представлен на рис. 1. Вершины соответствуют работам проекта, а стрелки показывают порядок их выполнения.

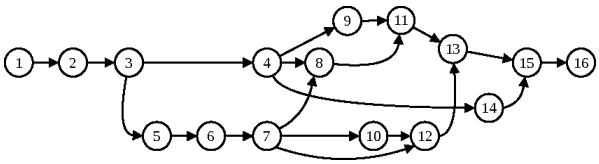

Рис. 1 – Сетевой график проекта разработки мобильного приложения

Модель содержит несколько параллельных ветвей. После разработки технического задания отдельно выполняются UX/UI-проектирование и проектирование базы данных. После создания серверной части и API становится возможной разработка клиентского приложения. Такая структура соответствует реальной организации работ в ИТ-проекте.

IV. Результаты расчёта

В результате прямого и обратного проходов по сетевой модели были рассчитаны ранние и поздние

сроки начала работ. Минимальная длительность проекта составила 76 дней. Критический путь имеет вид:

$$1 \to 2 \to 3 \to 5 \to 6 \to 7 \to 8 \to 11 \to 13 \to 15 \to 16$$

Данный путь включает ключевые этапы: инициацию проекта, анализ рынка, формирование технического задания, проектирование базы данных, разработку серверной части, создание API, разработку клиентского приложения, функциональное тестирование, устранение дефектов, публикацию и официальный запуск.

Работы критического пути не имеют резерва времени, поэтому задержка любой из них увеличивает общий срок проекта. Такие работы должны находиться под постоянным контролем руководителя проекта и выполняться основной командой разработки.

Для некритических работ получены положительные резервы времени. К ним относятся UX/UI-прототипирование, подготовка графического контента, настройка облачной инфраструктуры, аудит безопасности API и подготовка юридической документации. Наибольший резерв имеет подготовка юридической документации, что делает её удобным кандидатом для выполнения внешним исполнителем.

V. Управленческая интерпретация

Полученные результаты могут использоваться не только для оценки сроков, но и для принятия организационных решений. Критический путь показывает, какие работы определяют длительность проекта. Следовательно, именно эти этапы целесообразно закрепить за основной командой компании, так как они требуют постоянной коммуникации, быстрого согласования и максимального контроля.

Некритические работы с положительными резервами времени могут быть частично переданы на аутсорсинг. Например, внешним исполнителям можно поручить подготовку графического контента, юридической документации, отдельных UX/UI-материалов или аудит безопасности API. При соблюдении контрольных сроков это не увеличит общую длительность проекта, но позволит разгрузить основную команду и сосредоточить её ресурсы на серверной разработке, создании API, мобильной разработке, тестировании и исправлении дефектов.

Таким образом, сетевая модель позволяет не только определить продолжительность проекта, но и обосновать, какие работы следует выполнять внутри компании, а какие допустимо передать сторонним исполнителям без риска срыва календарного плана.

Выводы

В работе построена и проанализирована сетевая модель проекта разработки мобильного приложения для Android и iOS. Выполнен расчёт ранних и поздних сроков начала работ, определены резервы времени и найден критический путь.

Минимальная длительность проекта составила 76 дней. Критический путь включает работы, связанные с анализом, формированием требований, серверной и клиентской разработкой, тестированием, исправлением ошибок и публикацией приложения. Эти этапы требуют первоочередного контроля со стороны руководителя проекта.

Показано, что некритические работы с положительными резервами времени могут передаваться на аутсорсинг. Это позволяет разгрузить основную команду без увеличения общей продолжительности проекта.

Список литературы

1. Новицкий, Н. И. Сетевое планирование и управление производством / Н. И. Новицкий. – Минск: Новое знание, 2004. – 159 с.
2. Sommerville, I. Software Engineering / I. Sommerville. – 10th ed. – Pearson, 2016. – 796 p.
3. Pressman, R. S. Software Engineering: A Practitioner's Approach / R. S. Pressman, B. R. Maxim. – 9th ed. – New York: McGraw-Hill Education, 2019.

Ковалев Дмитрий Николаевич, студент факультета информационных технологий и управления Белорусского государственного университета информатики и радиоэлектроники, dimonkov2017@gmail.com.

Научный руководитель: Ломако Александр Викторович, доцент кафедры информационных технологий автоматизированных систем Белорусского государственного университета информатики и радиоэлекстроники, кандидат технических наук, доцент, lavlot@bsuir.by.