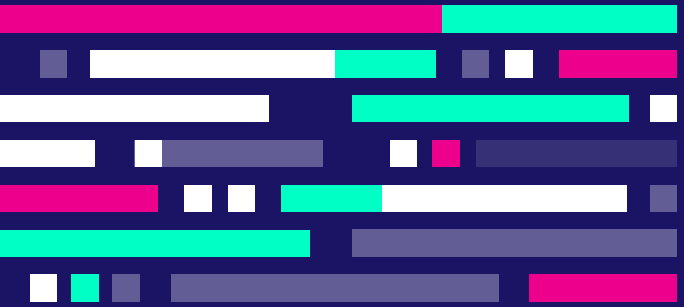
МЕТОДОЛОГИИ РАЗРАБОТКИ ПО

AGILE

Основные идеи Agile:



- люди и взаимодействие важнее процессов и инструментов;
- работающий продукт важнее исчерпывающей документации;
- сотрудничество с заказчиком важнее согласования условий контракта;
- готовность к изменениям важнее следования первоначальному плану.

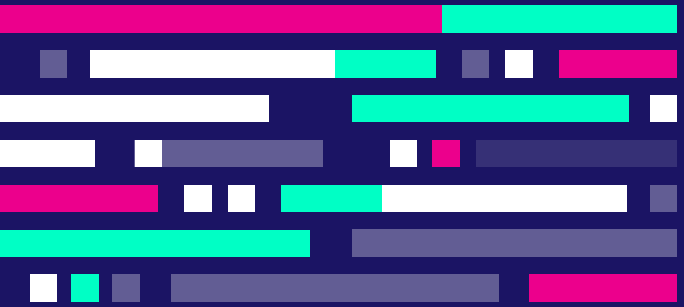
AGILE

Один из принципов - взаимодействие - подразумевает, что заказчик взаимодействует с командой, команда с заказчиком - все между собой. Это позволяет обмениваться опытом между участниками команды и клиентом и участвовать каждому из них в принятие решений. За счет такого подхода снижаются риски потери времени и денег и повышается способность команды решать сложные нестандартные задачи с высокой степенью неопределенности.

Однако взаимодействия всех и со всеми может вылиться в хаос, что влияет на все сферы разработки. Поэтому используя Agile нужно понимать ограничения: команды должны быть небольшие, участники должны быть компетентные и мотивированные, итерации короткие с максимально понятными целями, установлены четкие ограничения по времени и конечный результат должен быть очевидным.

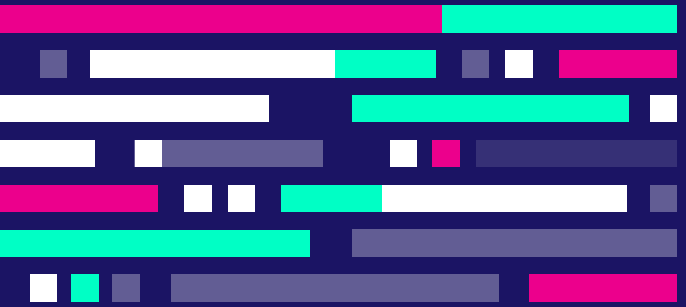
Agile прекрасно управляется с неопределенностью, предопределяя будущее на более короткий период. Правило такое: чем выше неопределенность - тем короче итерация, причем ее длительность может быть даже в рамках 24 часов, как и происходит на хакатонах. Вначале каждой итерации неизбежно выполняется контроль, ретроспектива, оценка и анализ результатов,планирование следующей итерации.

SCRUM



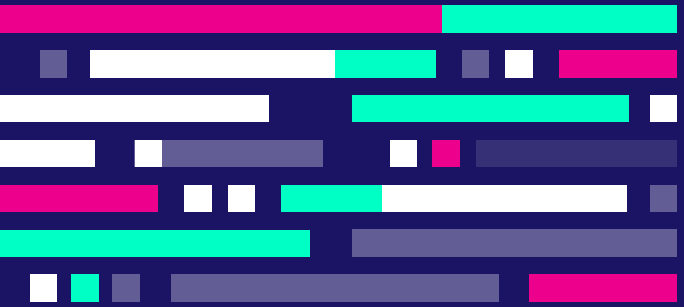
Scrum методология основывается на понятии спринта (sprint), в течении которого выполняется работа над продуктом. Перед началом каждого спринта проводится планирование (Sprint Planning), на котором производится оценка содержимого списка задач по развитию продукта (Product Backlog) и формирование бэклога на спринт (Sprint Backlog), в рамках которых и действует команда. Для спринта всегда существуют ограничения по времени, обычно от недели до месяца. Жизнь продукта таким образом разбита на равные по продолжительности спринты.

KANBAN



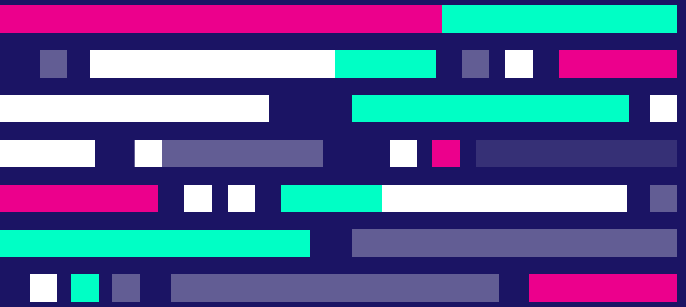
По Kanban методологии проект делится на этапы, которые визуализируются в виде канбан-доски. Этапы, это например: планирование, разработка, тестирование, релиз и т.п. Задачи в виде “карточек” перемещаются с этапа на этап. Новые задачи можно добавлять в любое время. Задача закрывается не по истечению конкретного времени, а по смене статуса на “завершено”. Канбан это методология из концепции “бережливого производства”.

RUP



RUP (Rational Unified Process) – разработка продукта при данном методе состоит из четырех фаз (начальная стадия, уточнение, построение, внедрение), каждая из которых включает в себя одну или несколько итераций. RUP огромная методология, которую трудно уложить в абзац текста, но методы, рекомендуемые RUP основаны на статистике коммерчески успешных проектов.

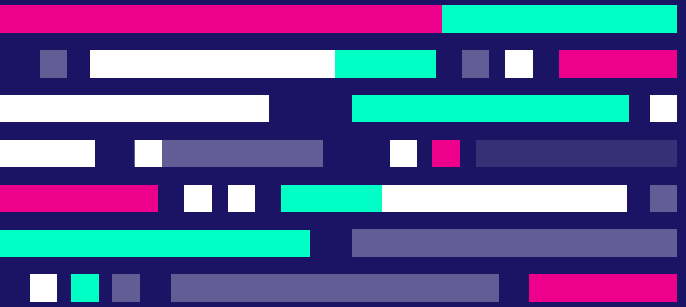
DSDM



DSDM (Dynamic Systems Development Model) – методология, которая демонстрирует набор принципов, предопределенных типов ролей и техник.

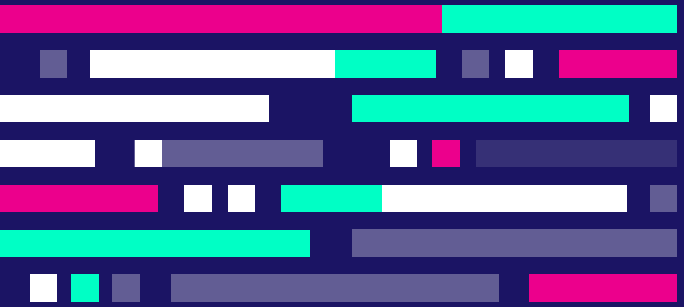
Принципы направлены на главную цель - сдать готовый проект вовремя и уложиться в бюджет, с возможностью регулировать требования во время разработки. DSDM входит в семейство гибкой методологии разработки программного обеспечения, а также разработок не входящих в сферу информационных технологий.

RAD



RAD (Rapid Application Development) – методология быстрой разработки приложений, которая предполагает применение инструментальных средств визуального моделирования (прототипирования) и разработки. RAD предусматривает небольшие команды разработки, сроки до 4 месяцев и активное привлечение заказчика с ранних этапов. Данная методология опирается на требования, но также существует возможность их изменений в период разработки системы. Такой подход позволяет сократить расходы и свести время разработки к минимуму.

XP



XP (Extreme Programming) – методология, которая ориентируется на постоянное изменение требований к продукту и предлагает 12 подходов для достижения эффективных результатов в подобных условиях. Среди них:

- Быстрый план и его постоянное изменение
- Простой дизайн архитектуры;
- Частое тестирование;
- Участие одновременно двух разработчиков в одной задаче или даже за одним рабочим местом;
- Постоянная интеграция и частые небольшие релизы.