Research and Development Company

634045, Россия, г. Томск,
ул. Красноармейская, 146, оф. 801

www.intecgroup.ru
office@intecgroup.ru

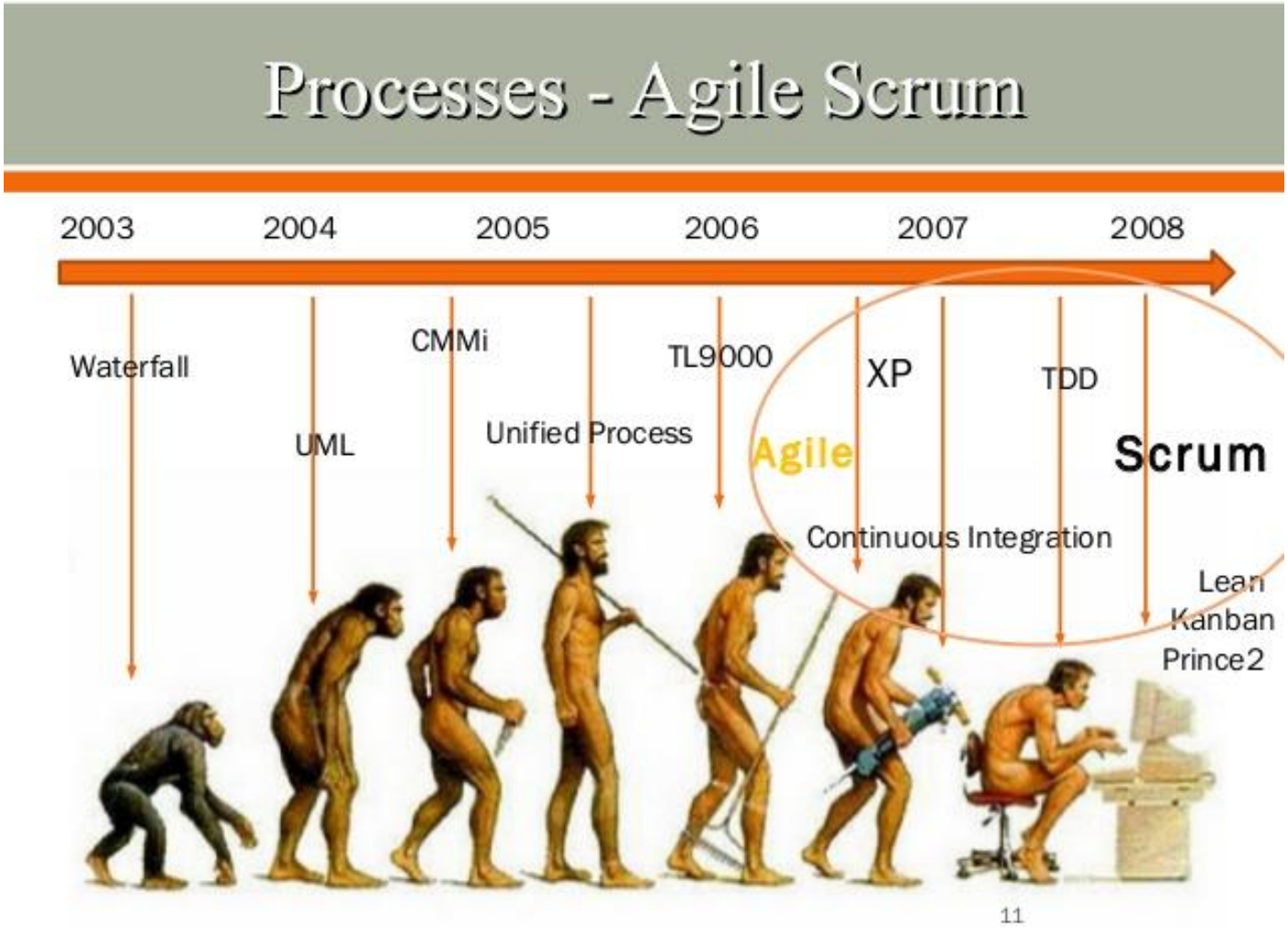
Методы управления командой

Гибкая методология разработки

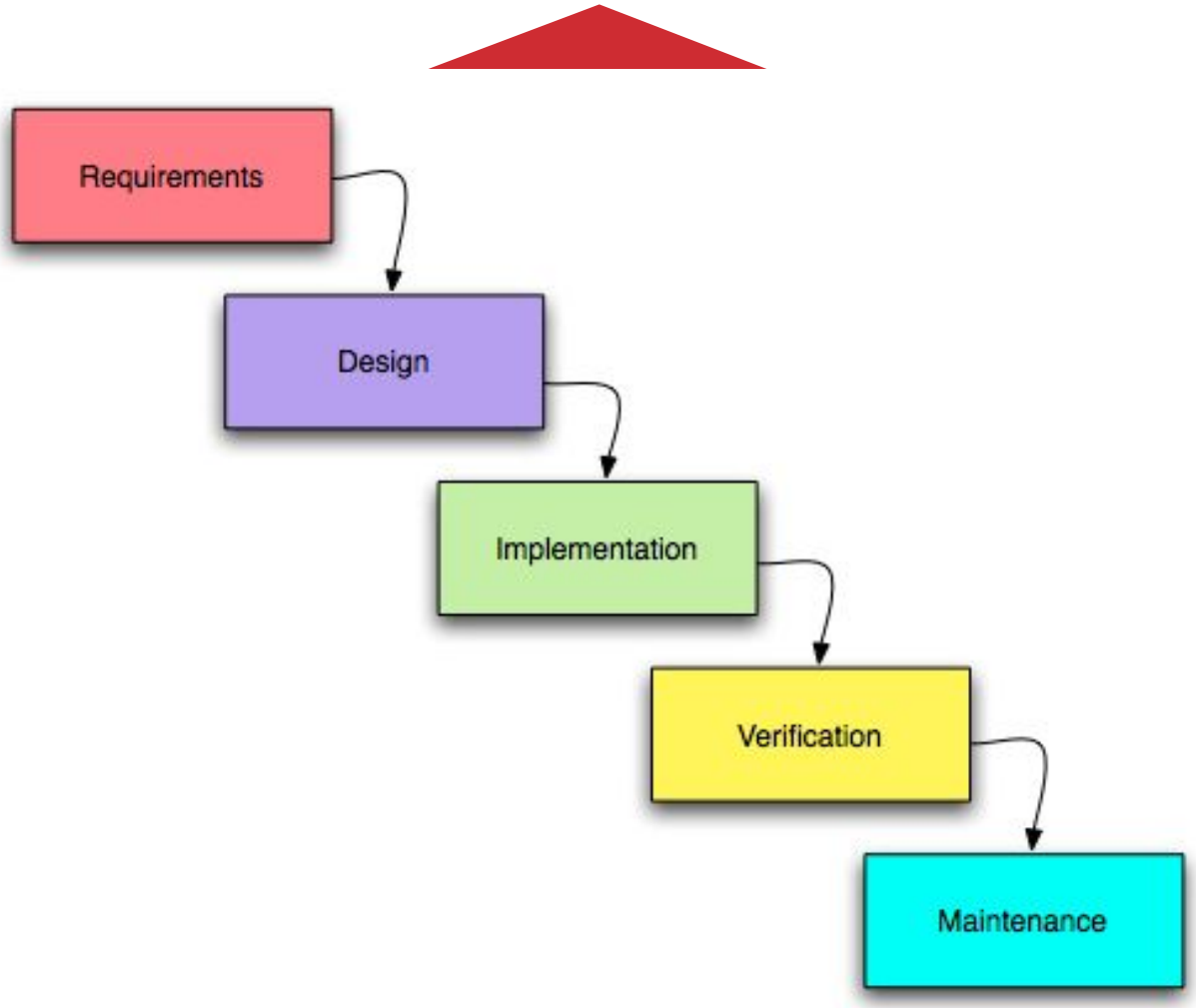
Гибкая методология разработки (англ. *Agile software development*, *agile-методы*) — серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля. Существует множество методик, относящихся к классу гибких методологий разработки, в частности экстремальное программирование, DSDM, Scrum, FDD и др.

- минимизация рисков путём сведения разработки к серии коротких циклов, называемых итерациями;
- итерации длятся две-три недели;
- каждая итерация сама по себе выглядит как программный проект в миниатюре и включает все задачи: планирование, анализ требований, проектирование, программирование, тестирование и документирование;
- подразумевается, что гибкий программный проект готов к выпуску в конце каждой итерации;
- по окончании каждой итерации команда выполняет переоценку приоритетов разработки.

История


Processes - Agile Scrum
2003
2004
2005
2006
2007
2008
Waterfall
UML
CMMi
Unified Process
TL9000
Agile
XP
TDD
Scrum
Continuous Integration
Lean
Kanban
Prince2

Waterfall


Requirements
Design
Implementation
Verification
Maintenance

Waterfall

1. Определение требований
2. Проектирование
3. Конструирование (также «реализация» либо «кодирование»)
4. Тестирование и отладка (также «верификация»)
5. Инсталляция
6. Поддержка

Основные идеи Agile

1. люди и взаимодействие важнее процессов и инструментов
2. работающий продукт важнее исчерпывающей документации
3. сотрудничество с заказчиком важнее согласования условий контрак
4. готовность к изменениям важнее следования первоначальному плану

Принципы Agile Manifesto

1. удовлетворение клиента за счёт ранней и бесперебойной поставки ценного программного обеспечения;
2. приветствие изменений требований даже в конце разработки (это может повысить конкурентоспособность полученного продукта);
3. частая поставка рабочего программного обеспечения (каждый месяц или неделю или ещё чаще);
4. тесное, ежедневное общение заказчика с разработчиками на протяжении всего проекта;
5. проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием;
6. рекомендуемый метод передачи информации — личный разговор (лицом к лицу);
7. работающее программное обеспечение — лучший измеритель прогресса;
8. спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределённый срок;
9. постоянное внимание улучшению технического мастерства и удобному дизайну;
10. простота — искусство не делать лишней работы;
11. лучшие технические требования, дизайн и архитектура получаются у самоорганизованной команды;
12. постоянная адаптация к изменяющимся обстоятельствам.

Скрам

- это набор принципов, на которых строится процесс разработки, позволяющий в жёстко фиксированные и небольшие по времени итерации, называемые спринтами (sprints), предоставлять конечному пользователю работающее ПО с новыми возможностями, для которых определён наибольший приоритет. Возможности ПО к реализации в очередном спринте определяются в начале спринта на этапе планирования и не могут изменяться на всём его протяжении. При этом строго фиксированная небольшая длительность спринта придаёт процессу разработки предсказуемость и гибкость.

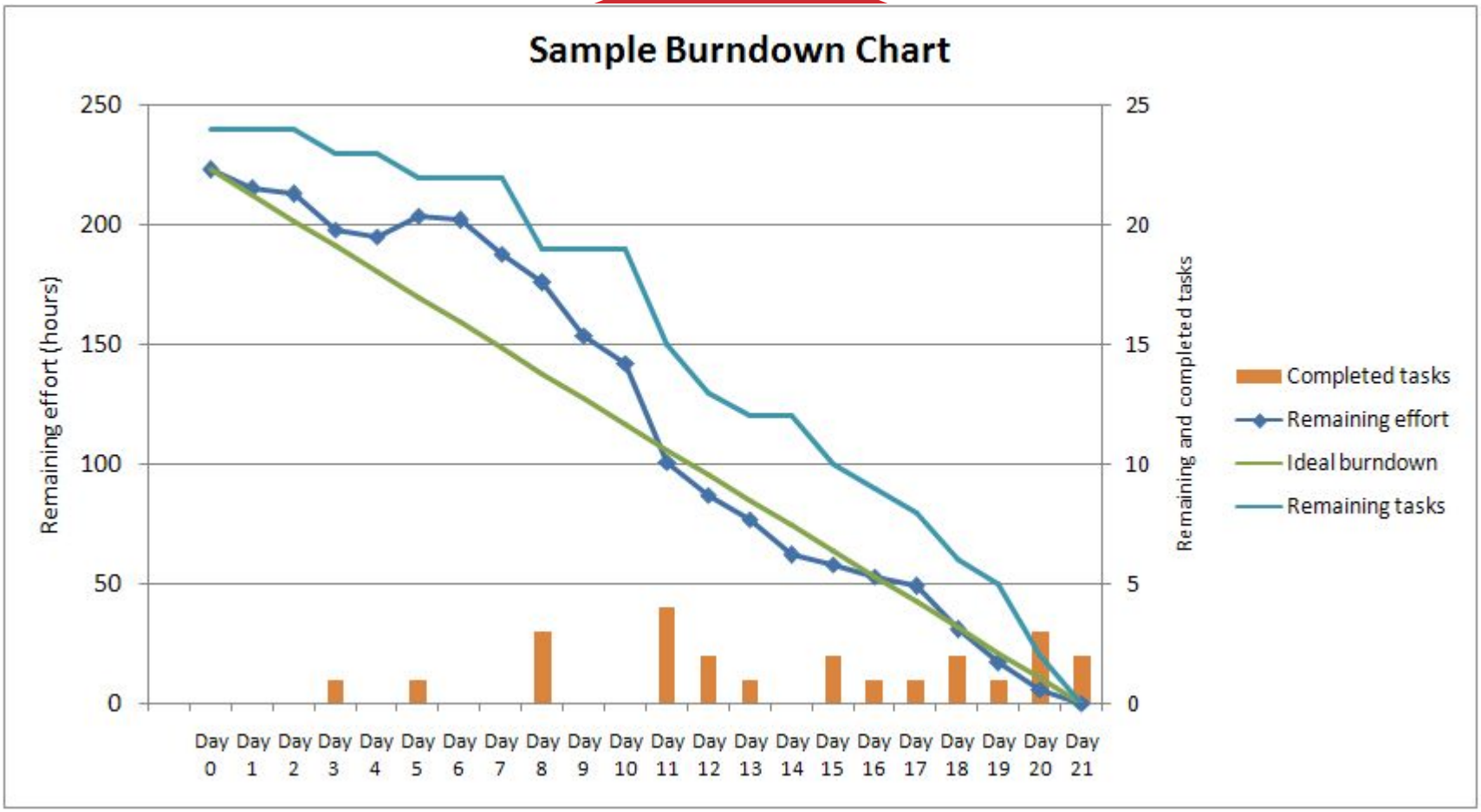
Спринт

- Это итерация в скраме, в ходе которой создаётся функциональный рост программного обеспечения. Жёстко фиксирован по времени. Длительность одного спринта от 2 до 4 недель. Разные команды подбирают длину спринта согласно специфике своей работы, составу команд и требований, часто методом проб и ошибок. Для оценки объёма работ в спринте можно использовать предварительную оценку, измеряемую в очках истории. Предварительная оценка фиксируется в бэклоге проекта.

Список пожеланий

- это список требований к функциональности, упорядоченный по их степени важности, подлежащих реализации. Элементы этого списка называются пользовательскими историями (user story) или элементами беклога (backlog items). Журнал пожеланий проекта открыт для редактирования для всех участников скрам-процесса.

Диаграмма сгорания



Задачи спринта

Скрам-мастер (Scrum Master)

Проводит совещания (Scrum meetings) следит за соблюдением всех принципов скрам, разрешает противоречия и защищает команду от отвлекающих факторов. Данная роль не предполагает ничего иного, кроме корректного ведения скрам-процесса. Руководитель проекта скорее относится к владельцу проекта и не должен фигурировать в качестве скрам-мастера.

Владелец продукта (Product Owner)

Представляет интересы конечных пользователей и других заинтересованных в продукте сторон. Является основным источником и контроллером User Stories.

Скрам-команда (Scrum Team)

кросс-функциональная команда разработчиков проекта, состоящая из специалистов разных профилей: тестировщиков, архитекторов, аналитиков, программистов и т. д. Размер команды в идеале составляет от 3 до 9 человек. Команда является единственным полностью вовлечённым участником разработки и отвечает за результат как единое целое. Никто, кроме команды не может вмешиваться в процесс разработки на протяжении спринта.

Минусы Scrum

1. Что делать с длинными проектами? Дорожная карта?
2. Оценка всего проекта?
3. Работает и ладно?
4. Общение? А когда работать то?

Kanban vs Scrum

Scrum	Kanban
Обязательны ограниченные по времени итерации	**Ограниченные по времени итерации необязательны.** Могут быть отдельные ритмы для планирования, выпуска и усовершенствования процессов. Также могут быть событийно-управляемые итерации вместо ограниченных по времени
Команда обязуется выполнить конкретный объем работы за эту итерацию	**Обязательства опциональны**
Как основная метрика для планирования и улучшения процессов используется **производительность**	Как основная метрика для планирования и улучшения процессов используется **время выполнения задачи**
Кросс-функциональные команды обязательны	Кросс-функциональные команды, опциональны. **Допустимы узкопрофильные команды**
Задачи должны быть разбиты на более мелкие так, чтобы они были	Нет каких-либо определенных размеров задач

Минусы Kanban

1. Кому нужна оценка?
2. Взаимодействие функциональных специалистов?
3. Обязательства опциональны?
4. Встречи команды не обязательны?
5. Нет заранее определенных ролей?
6. Приоритеты?

Наш опыт и опыт наших коллег

- У вас нет необходимости выбирать!
- Вы можете взять лучшее из Скрам и Канбан, подстроив механизмы agile практик именно под ваши проекты по аналогии с конструктором LEGO.
- Пробуйте, экспериментируйте, адаптируйте и помните, при всех возможностях Скрам и Канбан - всего лишь инструменты.
- Проекты неразрывно связаны со стратегией и идеологией компании, правила управления проектами базируются на правилах управления задачами и процессами.
- Каждый проект уникален и должен рассматриваться уникально с точки зрения управления.