В роли конструкторского бюро выступают компиляторы.

Эволюция методологий управления программными продуктами:

1. Как получится - незамкнутая система управления, полное доверие техлидам, планировка неформальная, бюджет/сроки не оговорены.
2. Waterfall - жесткое управление с обратной связью, есть план проекта, выбор отклонений от норм, коррекция и возврат к начальной точке.
3. Гибкое управление
4. Метод частой поставки - самонаведение, уточнение целей

Во избежания проблем, управленец должен рационально распределять возможности и время программистов.

Модели разработки ПО

Квалифицируют по “весу” - кол-во формализованных процессов и детальность их регламентации. Чем больше таких процессов документировано, тем больше вес модели.

* ГОСТ-19 (единая система программной документации) - используется для госзаказов
* ГОСТ-34 (стандарты на разработку и поддержку автоматизированных систем)

ГОСТы ориентированны на последовательный подход к разработке ПО. Разработка происходит поэтапно. На каждом этапе строго определены работы. Проект закрывается с большим кол-вом формализованных и громоздких документов. Разработка ТЗ подталкивает к каскадной разработке.

**SW-CMM** (software capability maturity model) - модель зрелости возможностей (модель полноты потенциала) создания ПО: эволюционная модель развития способности компании разрабатывать программное обеспечение. Модель определяет пять уровней зрелости разработки ПО:

1. начальный - разработка хаотична, определено только несколько процессов, успех проекта зависит от конкретных исполнителей.
2. повторяющийся - определены основные процессы разработки (трекинг бюджета, сроков, функционала). Некоторые процессы упорядочены, они необходимы для повтора предыдущие успехи на аналогичных проектах.
3. определение - процесс разработки ПО и управления проектами, описанные и запущены в единую систему процессов компании.

—-вики——

1. Начальный. Самый примитивный статус организации. Организация способна разрабатывать ПО. Организация не имеет явно осознанного процесса, и качество продукта целиком определяется индивидуальными способностями разработчиков. Один проявляет инициативу, и команда следует его указаниям. Успех одного проекта не гарантирует успех другого. При завершении проекта не фиксируются данные о трудозатратах, расписании и качестве.
2. Повторяемый. В некоторой степени отслеживается процесс. Делаются записи о трудозатратах и планах. Функциональность каждого проекта описана в письменной форме. В середине 1999 года лишь 20 % организаций имели 2-й уровень или выше.
3. Установленный. Имеют определённый, документированный и установленный процесс работы, не зависящий от отдельных личностей. То есть вводятся согласованные профессиональные стандарты, а разработчики их выполняют. Такие организации в состоянии достаточно надёжно предсказывать затраты на проекты, аналогичные выполненным ранее.
4. Управляемый. Могут точно предсказать сроки и стоимость работ. Есть база данных накопленных измерений. Но нет изменений при появления новых технологий и парадигм.
5. Оптимизированный. Есть постоянно действующая процедура поиска и освоения новых и улучшенных методов и инструментов.

**RUP** - рациональный унифицированный процесс.

В основе RUP лежат следующие принципы:

* Ранняя идентификация и непрерывное (до окончания [проекта](https://ru.wikipedia.org/wiki/Проект)) устранение основных рисков.
* Концентрация на выполнении требований заказчиков к исполняемой программе (анализ и построение модели [прецедентов](https://ru.wikipedia.org/wiki/Прецедент_(UML)) (вариантов использования)).
* Ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки.
* [Компонентная архитектура](https://ru.wikipedia.org/wiki/Компонентная_архитектура), реализуемая и тестируемая на ранних стадиях проекта.
* Постоянное обеспечение качества на всех этапах разработки [проекта](https://ru.wikipedia.org/wiki/Проект) (продукта).
* Работа над проектом в сплочённой команде, ключевая роль в которой принадлежит архитекторам.

**MSF** - Microsoft Solutions Framework - методология разработки [программного обеспечения](https://ru.wikipedia.org/wiki/Программное_обеспечение), предложенная корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). MSF опирается на практический опыт Microsoft и описывает управление людьми и рабочими процессами в процессе разработки решения.

MSF представляет собой согласованный набор концепций, моделей и правил.

MSF содержит:

* **модели**:
  + модель проектной группы
  + модель процессов
* **дисциплины**:
  + дисциплина [*управление проектами*](https://ru.wikipedia.org/wiki/Управление_проектами)
  + дисциплина [*управление рисками*](https://ru.wikipedia.org/wiki/Управление_рисками)
  + дисциплина [*управление подготовкой*](https://ru.wikipedia.org/w/index.php?title=Управление_подготовкой&action=edit&redlink=1)

**Индивидуальный процесс разработки** ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *Personal software process*, *PSP*) — процесс разработки [ПО](https://ru.wikipedia.org/wiki/Программное_обеспечение), помогающий разработчикам понимать и улучшать собственную производительность. Создан для применения принципов [модели зрелости процессов](https://ru.wikipedia.org/wiki/Модель_зрелости_процессов) к практике одного разработчика.

Предоставляет разработчикам описания методов планирования и оценки, показывает как измерять собственную продуктивность и соотносить её с существующим планом.

Термины "Personal Software Process" и "**PSP**" зарегистрированы и принадлежат [Университет Карнеги — Меллон](https://ru.wikipedia.org/wiki/Университет_Карнеги_—_Меллон).

Один из основных аспектов PSP — использование накопленной статистики для анализа и улучшения показателей процесса разработки. Сбор статистики включает 4 элемента:

* Скрипты.
* Оценки. Включают 4 основных элемента:
* Размер — оценка размера для части продукта. Например, количество строк кода (LOC — Lines Of Code).
* Качество — количество ошибок в продукте.
* Усилия — оценка времени, требующегося для завершения задачи, обычно записываемое в минутах.
* Планирование — оценка хода проекта, перемещаемая между планируемыми и завершенными пунктами.
* Стандарты кодирования. Применение стандартов к процессу может обеспечить точные и постоянные данные.
* Формы.

Согласно требованиям программист должен уметь:

* учитывать время, потраченное на работу над проектом;
* учитывать найденный дефект, квалифицировать его тип;
* оценивать размер задачи;
* планировать программные задачи;
* составлять график работы;
* выполнять индивидуальную проверку кода;
* выполнять регрессивные тесты

**TSP** - team software process

Мемберы должны уметь:

* ставить цели
* поддерживать мотивацию
* планировать действия

Все эти модели позволяют достигнуть пятого уровня **CMM**.

**Выбор типа процессов**

Тяжелые и легкие модели имеют свои плюсы и минусы.