Управление проектом

при разработке программного обеспечения

Текущее состояние

- Сложность проектов растет
- Технологии стремительно развиваются
- Производственные ресурсы: люди
- **Управление как достижение *баланса* между:**
  - Теорией и практикой
  - Технологией и людьми
  - Стоимостью для заказчика и прибыльностью для производителя
  - Стратегией и тактикой
- **Необходимо обеспечить баланс между целями заинтересованных сторон**

Языки представления

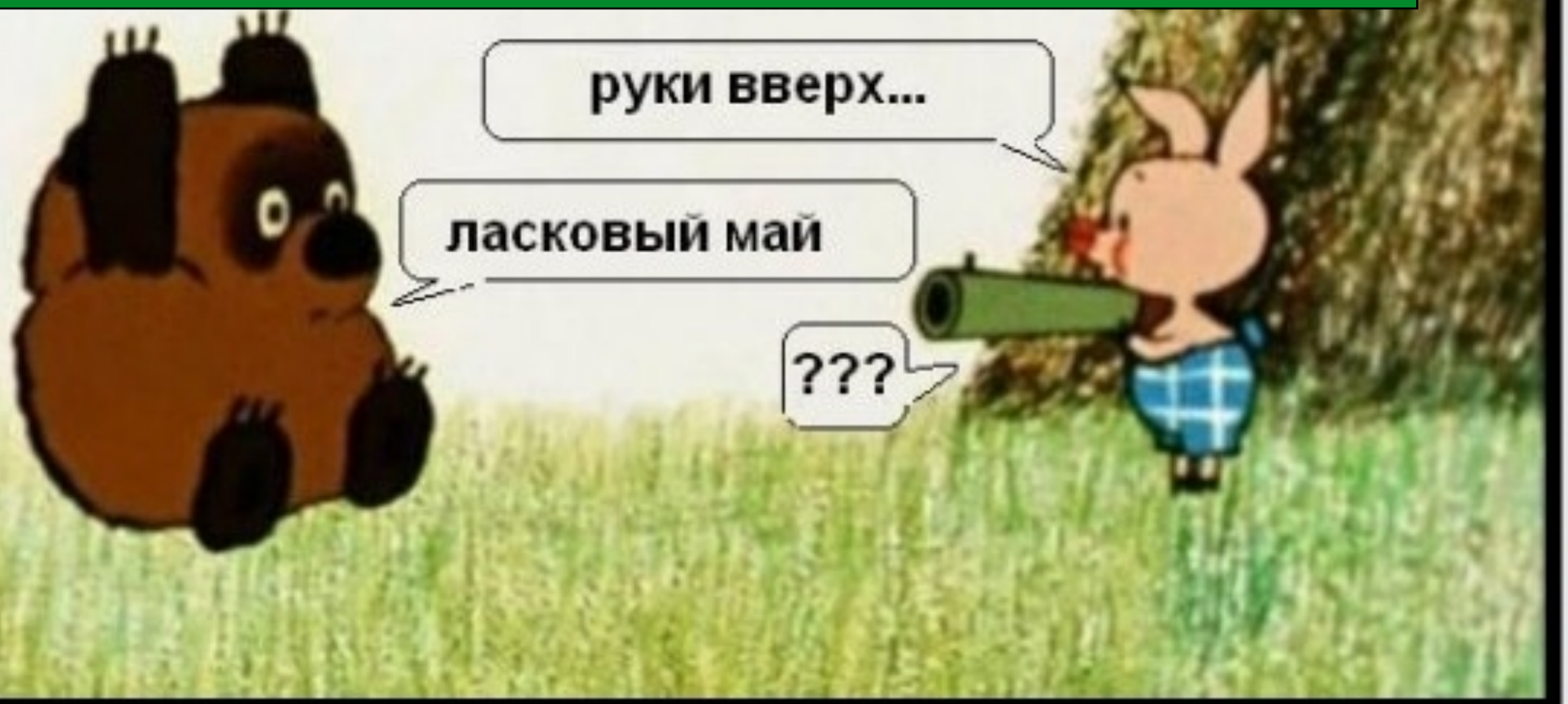
- **Языки представления в разработке ПО**:
  - Требования (язык проблемной области)
  - Проектные решения (языки, используемые разработчиками)
  - Реализация (язык, понимаемый компьютерами)

Управление

- **Управление является предметом рассуждений и принятия решений, зависящих от ситуации**
  - Для управления разработкой ПО не придумано поваренных книг
  - Не существует рецептов поиска заведомо правильных решений

Традиционное управление разработкой

- Разработка ПО по-прежнему **абсолютно непредсказуема!**
- **Управление определяет успех** или неудачу в большей степени, чем технологические преимущества!
- Огромное количество ПО переделывается или выбрасывается!



Каскадная модель

- **Два основных этапа**
  - Анализ
  - Кодирование
- **Рискованный, допускает неудачное завершение**
  - Стадия тестирования в конце разработки – первый момент, где можно сравнить характеристики системы с их значениями, установленными при анализе
  - Изменения могут стать разрушительными для системы
  - Требования могут оказаться не выполненными

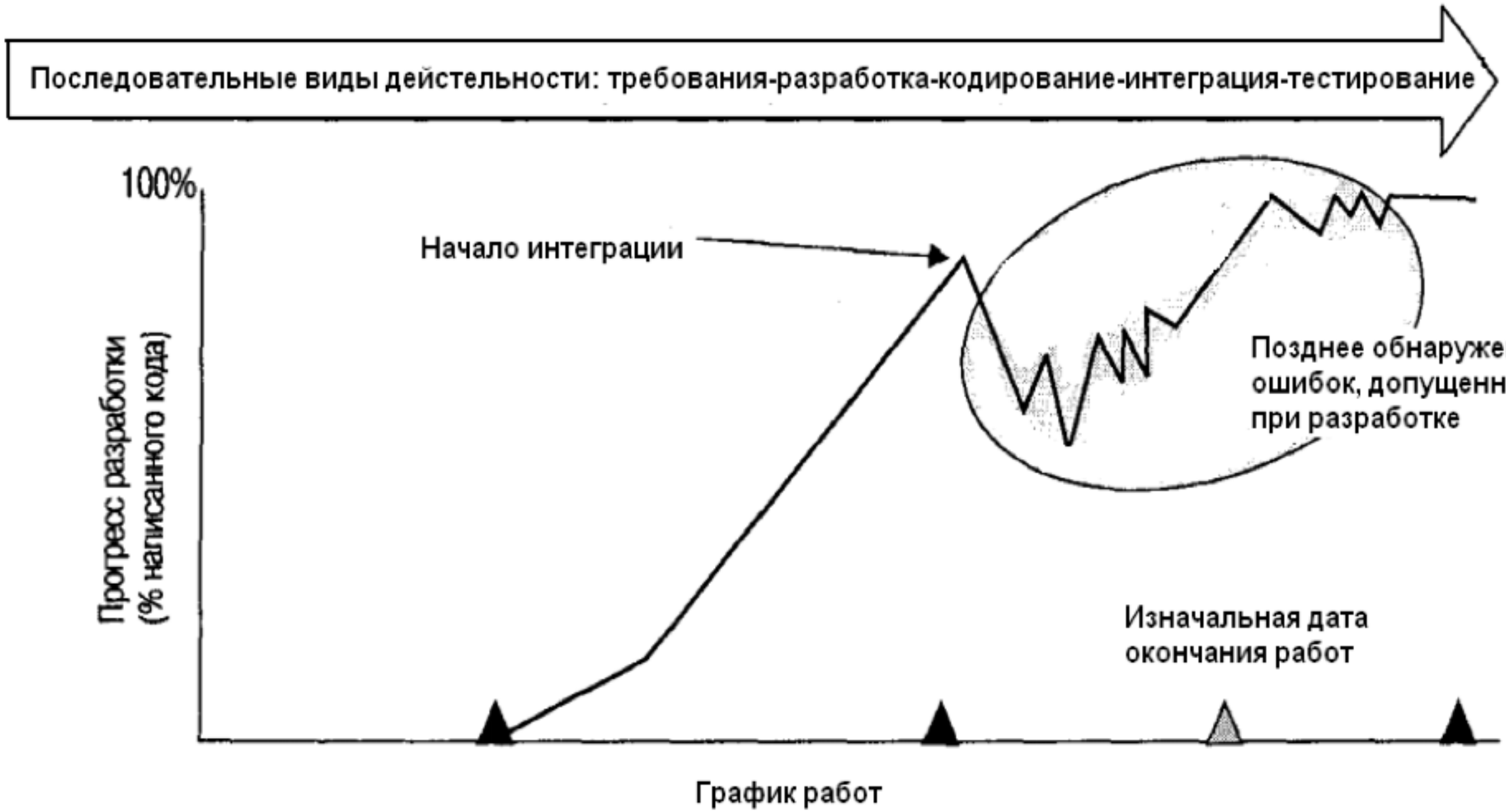
Традиционное управление

- *Традиционное управление = практика использования каскадной модели*
- «Узкие места» традиционного управления:
  - Поздние интеграция и обнаружение ошибок
  - Позднее разрешение рисков
  - Функциональная декомпозиция
  - Противостояние между участниками проекта
  - Чрезмерное внимание, уделяемое документации

Затянувшаяся интеграция

- Раннее завершение «бумажного» проектирования
- Переход к кодированию на поздних этапах
- Кошмарные трудности при интеграции
- Проблемы при реализации и неточности в интерфейсах
- Жестко регламентированные бюджет и время сдачи системы
- Использование неоптимальных решений (Нет времени на новое проектирование)
- Трудно сопровождаемый сырой продукт, передаваемый с опозданием

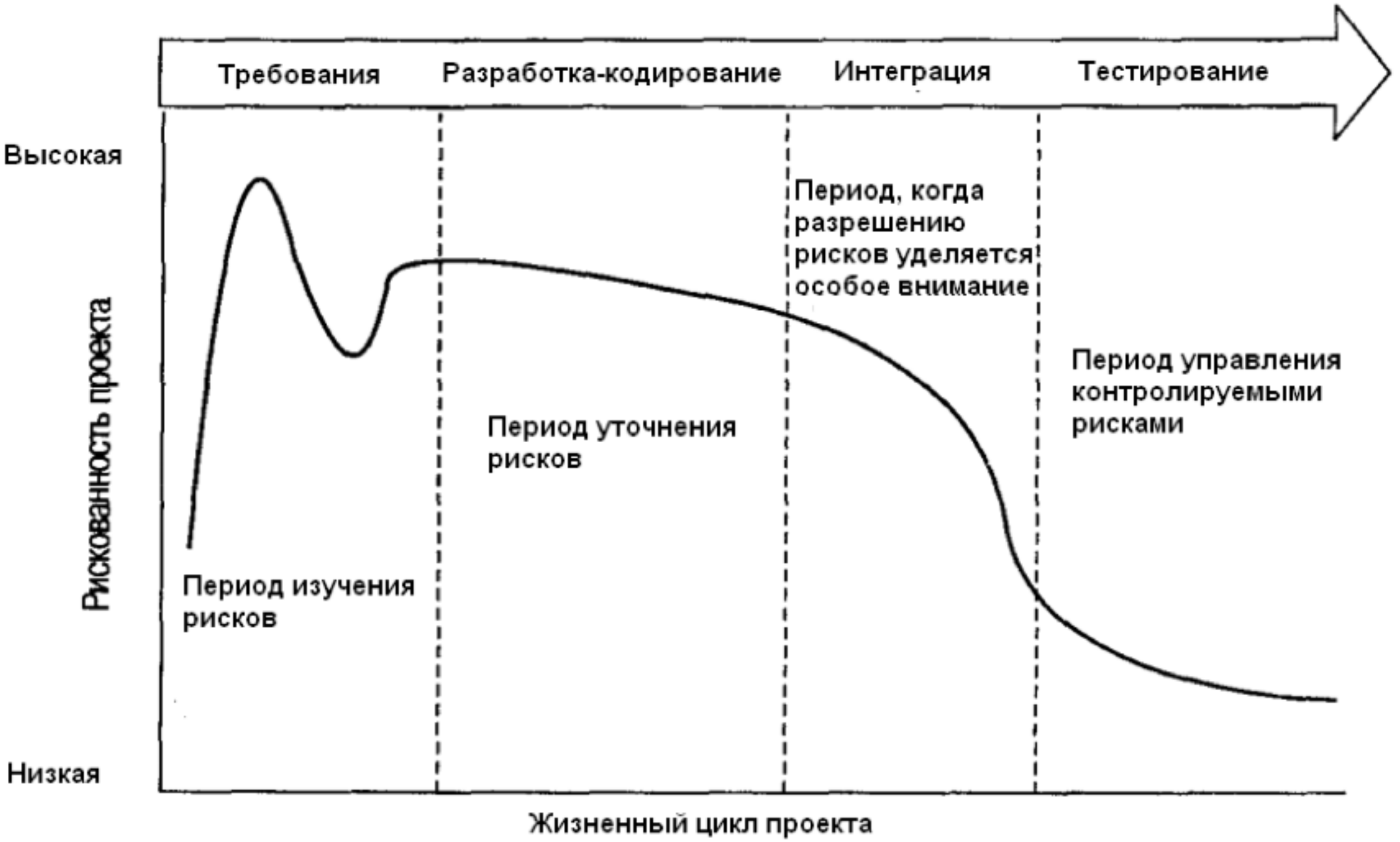
Традиционный процесс



Затраты в каскадной модели

Деятельность	Затраты
Менеджмент	5 %
Определение требований	5 %
Проектирование	10 %
Кодирование и тестирование модулей	30 %
Интеграция и тестирование	40%
Ввод в действие	5 %
Создание среды разработки	5 %
Всего:	100 %

Позднее разрешение рисков



Функциональная декомпозиция

- Требования специфицируются как функциональные требования
- Требования предъявляются к получившимся компонентам
- Закрепляется в контрактах и в декомпозиции работ
- Структура ПО организована в соответствии с определенной структурой требований

Противостояние между участниками

- Сложности в определении требований и обмене информацией в виде (бумажных) документов
- Отсутствие жестко заданной системы записей (шаблонов), наличие специальных форматов данных
- Одноразовый способ рассмотрения с необходимостью быстрого реагирования

Много совещаний и бумажной работы

- **Много документации для демонстрации прогресса**
- **Совещания для обмена мнениями**
  - Низкая ценность для участников
  - Высокая стоимость подготовки
  - Видимость прогресса
  - Недостаточное внимание самому продукту
- **Основные контрольные точки**
  - Обсуждались только конкретные документы
  - На их подготовку затрачивались усилия, которые лучше было бы потратратить на выполнение задач проекта

Эффективность традиционного управления

- Стоимость разработки и сопровождения = функция числа строк кода
- 80% работы выполняют 20% работающих
- Поиск и обнаружение ошибки при сдаче ПО обходятся в 100 раз дороже, чем на ранних стадиях разработки
- Можно сократить срок разработки не более, чем на 25% от номинального
- На каждый рубль, вложенный в разработку, тратим два на сопровождение

Экономика разработки

Ключевые замечания

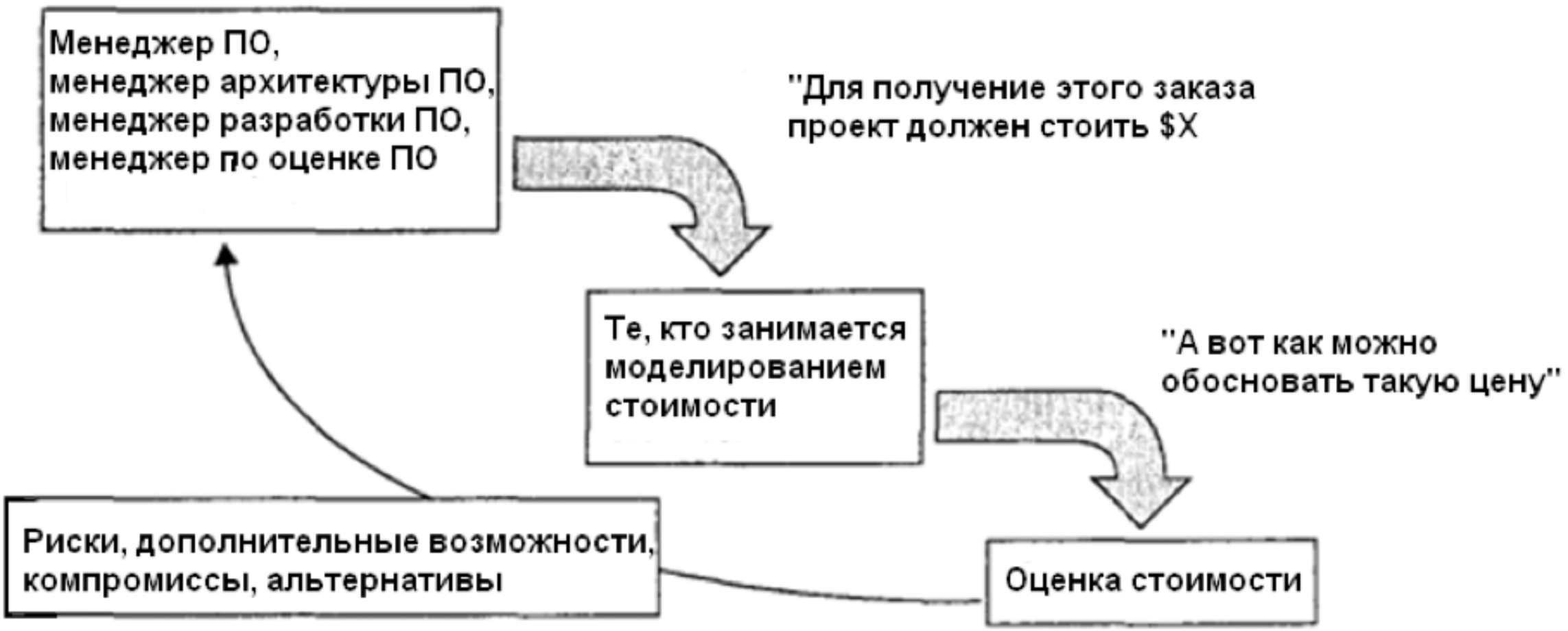
- Экономические результаты традиционных проектов - следствие преобладания разработки на заказ, узко специализированных процессов и платы за большой масштаб
- Современные модели стоимости основаны на эмпирических данных проектов
- Современная схема процесса позволяет бороться с основными причинами платы за большой масштаб

Модель определение стоимости

- Размер конечного продукта
- Особенности процесса
- Возможности персонала
- Используемая среда разработки
- Требуемое качество продукта

$$\text{Трудоемкость} = (\text{Размер}^{\text{Процесс}})(\text{Персонал})(\text{Среда})(\text{Качество})$$

Обоснование цены
Часто делают так :(



Совершенствование экономики разработки ПО

- Современная технология позволяет уменьшить количество кода, создаваемого человеком
- Современные процессы разработки являются итерационными
- Современная среда разработки и среда сопровождения позволяют автоматизировать процесс

Ключевые параметры модели стоимости

- Уменьшение ***размера*** или сложности ПО
- Усовершенствование ***процесса*** разработки
- Использование более квалифицированного ***персонала*** или хороших команд
- Использование лучшей ***среды*** (инструментария для автоматизации процесса)
- Достижение уступок и компромиссов для пороговых значений ***качества***

Уменьшение написания кода

- **Компонентно-ориентированная разработка!**
  - Языки программирования высокого уровня
  - Объектно-ориентированные методы и визуальное моделирование
  - Повторное использование
  - Коммерческие компоненты

Что такое процесс

- **Мета процесс (уровень бизнеса)**
  Политика, приемы и практика, которые присущи конкретной организации
- **Макропроцесс (уровень проекта)**
  Политика, приемы и практика, которые присущи конкретному проекту. Можно рассматривать как экземпляр мета процесса.
- **Микропроцесс (уровень итерации)**
  Политика, приемы и практика, которые присущие конкретной команде разработчиков.

Эффективность команды

- **Хорошо управляемый проект** может быть успешно выполнен обычной командой
- Плохо управляемый проект почти никогда не будет успешным, даже если над ним трудится группа разработчиков-экспертов
- **Система, архитектура которой разработана правильно**, может быть реализована и обычной командой разработчиков
- Система с плохой архитектурой заставит путаться даже команду разработчиков-экспертов.

Принципы формирования команды (Barry W. Boehm)

- **Принцип верховенства таланта** – меньшее количество лучших специалистов
- **Принцип соответствия заданий** – подбирайте задачи в соответствии с навыками и мотивацией имеющихся специалистов
- **Принцип карьерного роста** – помочь найти себя
- **Принцип командного равновесия** – люди дополняют и гармонично сочетаются друг с другом
- **Принцип исключения** – человек, выпадающий из команды, никому не принесет пользы

Качества менеджера проекта

- **Качества лидера команды**
  - *Умение нанимать*
  - *Умение взаимодействовать с заказчиком*
  - *Умение принимать решения*
  - *Умение создавать команду*
  - *Умение убеждать*

Достижение качества

- Экспертные оценки как второстепенные механизмы
- На ранних стадиях разработки уделить особое внимание критичным вариантам использования
- Использовать метрики и показатели измерения прогресса и качества архитектуры
- Обеспечить поддержку среды на протяжении всего жизненного цикла разработки
- Использовать визуальное моделирование, повторное использование и самодокументирование
- Решать проблемы производительности на основе оценок, полученных при демонстрациях

Традиционные принципы программной инженерии

1. Достигайте наивысшего качества
2. Передавайте продукты заказчику как можно раньше
3. Оценивайте альтернативы при разработке
4. Используйте различные языки на разных стадиях
5. Минимизируйте семантический разрыв
6. Ставьте методы выше инструментов
7. Хорошее управление более важно, чем хорошая технология
8. Разрабатывайте так, чтобы можно было вносить изменения
9. Знайте, что энтропия ПО возрастает

Современные принципы

1. Процесс на упреждающей разработке архитектуры
2. Итерационный, управляемый рисками, процесс
3. Разработка, основанная на компонентах
4. Среда для управления изменениями
5. Поддержка цикличной разработки (преобразований)
6. Объективный контроль качества и оценка прогресса
7. Подход, основанный на демонстрациях
8. Планирование с возрастающим уровнем детализации
9. Конфигурационное управление

Решение проблем традиционного подхода

Проблемы	Решения
Позднее обнаружение дефектов, доработки	Процесс, противостоящий рискам
Технологические проблемы	Упреждающая разработка архитектуры
Антагонизм между заинтересованными лицами	Подход, основанный на демонстрациях
«Ползучее» изменение требований	Итерационная разработка, варианты использования
Ненадлежащая производительность	Оценка, основанная на демонстрациях
Ненадлежащее функционирование	Итерационная разработка, ранние прототипы

Итерационный процесс

Модель COCOMO II (Процесс = 1.01 – 1.26)

- **Наличие прецедентов приложений**
  - Итерационный процесс
- **Гибкость процесса**
  - Управление изменениями
  - Настраиваемый процесс
- **Разрешение рисков, связанных с архитектурой**
  - Упреждающая разработка архитектуры
  - Подход, основанный на демонстрациях
- **Сплоченность команды**
  - Подход, основанный на моделях
  - Цикличная разработка
- **Зрелость процесса создания ПО**
  - Объективный контроль качества