
# Первый опыт внедрения WPF в сложной системе (C++ и COM)

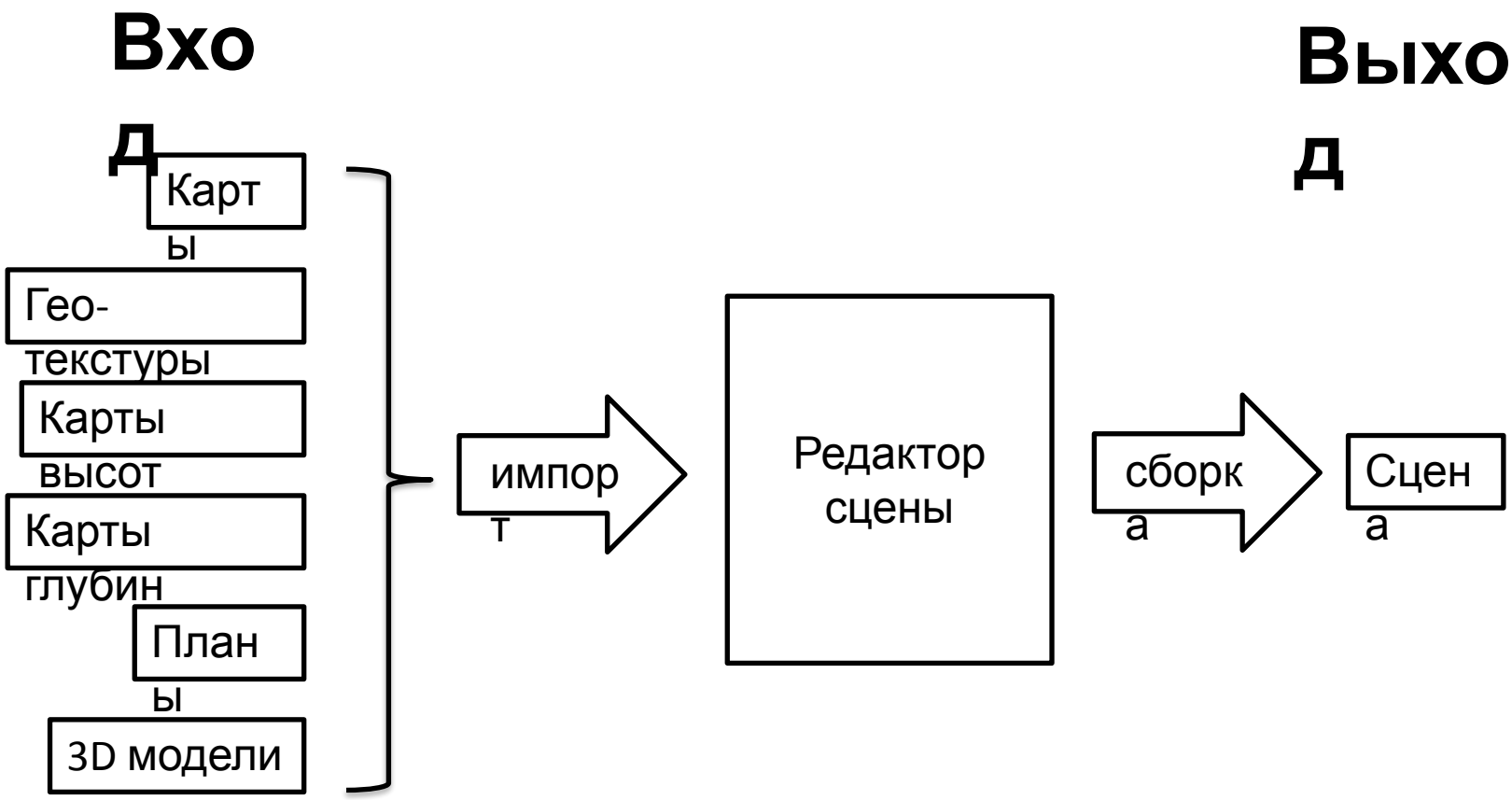
Михаил Павлов

Transas

# Цели доклада

- рассказать о проблемах на начальных  этапах внедрения WPF
- сформулировать рекомендации повышения эффективности разработки

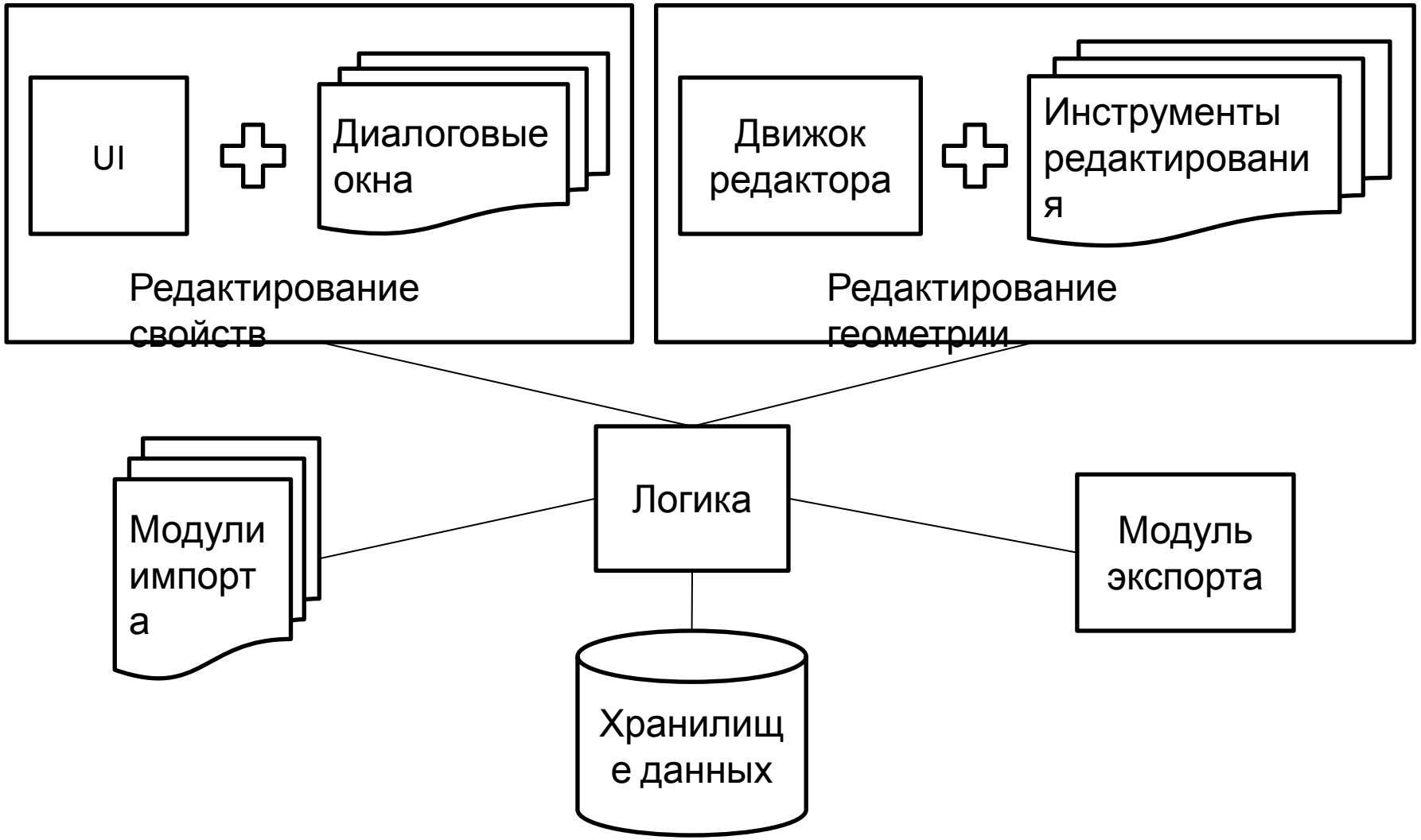
# Задачи продукта



# Сцена


20( 19) FPS
2084 DIPs
317.3 Video Memory Mb
413972 Tris
400Kb LUA Memory
284.67sec World time

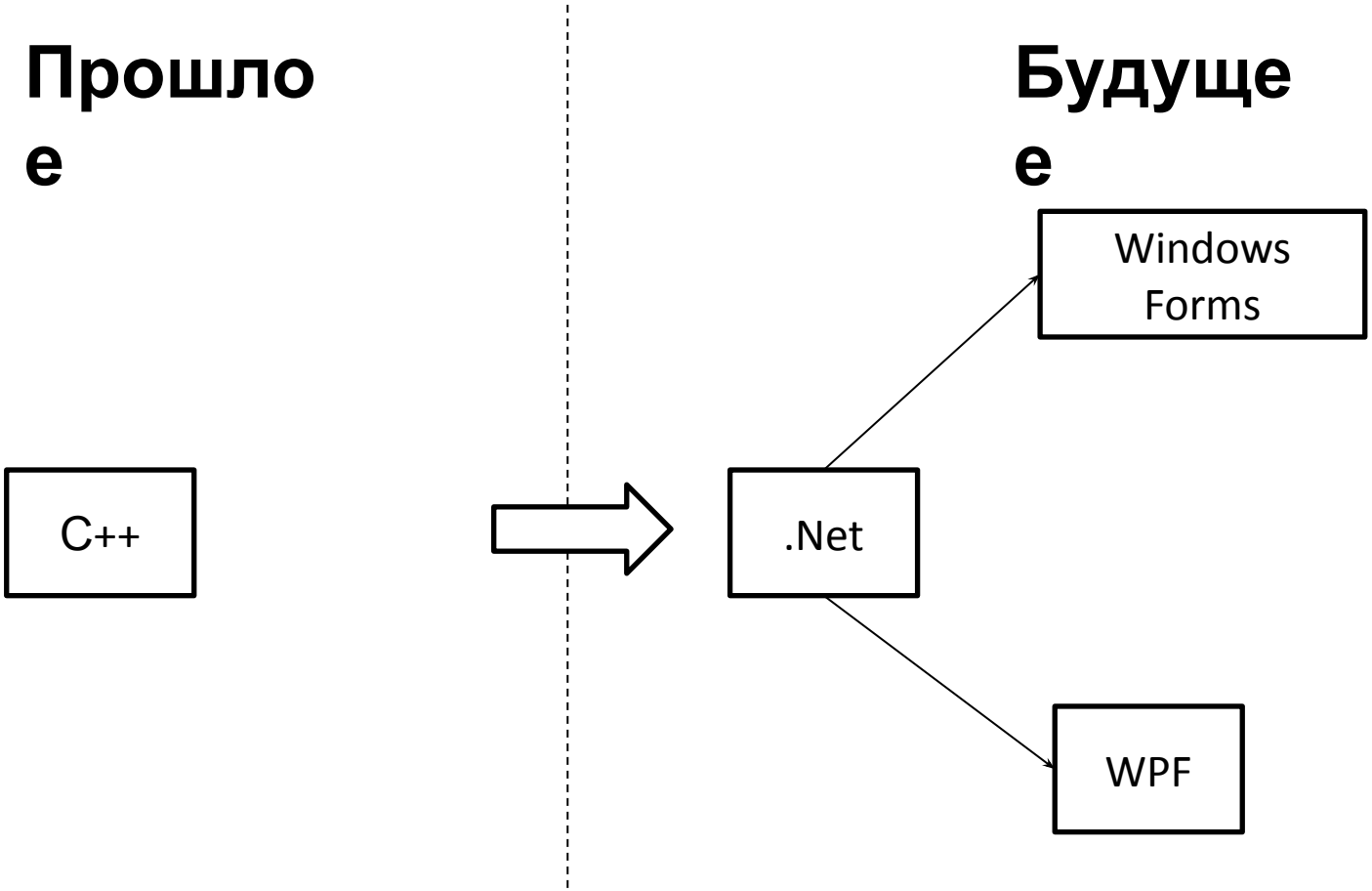
# Основные части



# Технологии

- Логика – C++, COM
- UI – MFC, ATL, WTL, C#(WF)
- Движок редактора – C++, MFC, GDI+
- Модули импорта-экспорта – C++, COM
- Визуализация – C++, COM, OpenGL
- Прочие модули – C++, COM, ATL, WTL, C#

# Варианты развития



# Проблемы

- Устаревший дизайн
- Падение скорости разработки UI
- Ограничения в расширяемости
- Отставание в технологиях

# Ожидаемые плюсы

- Переход на новейшие технологии
- Программист пишет только код
- Дизайном занимаются дизайнеры
- Улучшение внешнего вида
- Сложные проблемно-ориентированные компонент UI
- Ускорение разработки UI
- Использование скинов

**Необходимое условие: поддержка использования .Net на уровне ядра системы**

# Причины отказа от COM

- Много сопутствующего кода
- Проблемы синхронизации Interop оберток
- Потери быстродействия
- Не везде поиск ошибок во время компиляции

# Тестирование на изолированной утилите

# Первые впечатления

- Разработка интерфейса в стиле WF на WPF менее эффективна, чем на WF
- Легкости модификации системы при внесении изменений, нет и в помине
- Дизайн окон вручную съедает неоправданно много времени

# Коррекция разработки

- Использовать Binding совместно с моделью Data-Model-View
- Expression Blend в качестве редактора дизайна UI
- Разделить обязанности между дизайнером и программистом
- Увеличить количество разработчиков UI до двух человек.

# Результат коррекции:

# катастрофическое падение скорости разработки :[]

# Причины падения скорости

- Требуется переосмысление архитектуры
- Множество корректур дизайна
- Замусоренный код от дизайнера
- Формирование библиотеки стилей
- Формирование базового функционала
- Переход на векторную графику
- Тонкости использования WPF
- Недоработки библиотеке WPF

# Текущее положение вещей

# Практический опыт

# => экономия времени

# Бюрократия

# => упорядочивание внесения изменений

# Баланс обязанностей

# => экономия времени

# Сформулированы пожелания заказчика

# => снижение потока изменений

# Баланс между переделкой и повторным использованием графических ресурсов

=> экономия времени

# Мы перешли на WPF :)

# Оставшиеся проблемы

- Наследие прошлого
- Правильная интеграция 3D визуализации
- Перевод всего приложения на WPF

# Ожидания и реальность

# Переход на новейшие технологии не до конца

**Разделение обязанностей дизайнера и программиста – в небольшом объеме**

**Улучшение внешнего вида –**
**однозначно да**

# Повышение функциональности интерфейса – да, но с оговорками

# Скины –
# автоматически (by design)

# Итоговая скорость разработки

WTL < MFC < WPF< WF

Без учета затрат на дизайн:

- для окон средней сложности WPF= WF
- для окон большой сложности WF < WPF

# Рекомендации

# Внедрять WPF должны программисты .Net (квалификация)

**Внедрять должны минимум два программиста**
**(совещательность)**

**Для одного из программистов желателен опыт работы с WPF (центр кристаллизации знаний)**

**Для дизайнера желателен опыт верстки HTML (подобие)**

**Структурируйте разработку UI с целью упорядочивания внесения изменений и понимания остальными происходящего (экономия времени и нервов)**

**Начинайте разработку с простой задачи с акцентом на библиотеку стилей (задел в ширину)**

**Продолжайте разработку с самой сложной, но локальной задачи (задел архитектуры)**

**Помните – архитектура главное, остальное по нескольку раз меняется (акцент)**

# Спасибо за внимание

Михаил Павлов

Transas

pavmb2001@mail.ru