www.dpi.solutions

clang как инструмент парсинга и кодогенерации для C++

Антон Наумович
DPI.Solutions / LogicNow

www.dpi.solutions

Чем я занимаюсь

Антон Наумович

Более 12 лет опыта в разработке и менеджменте

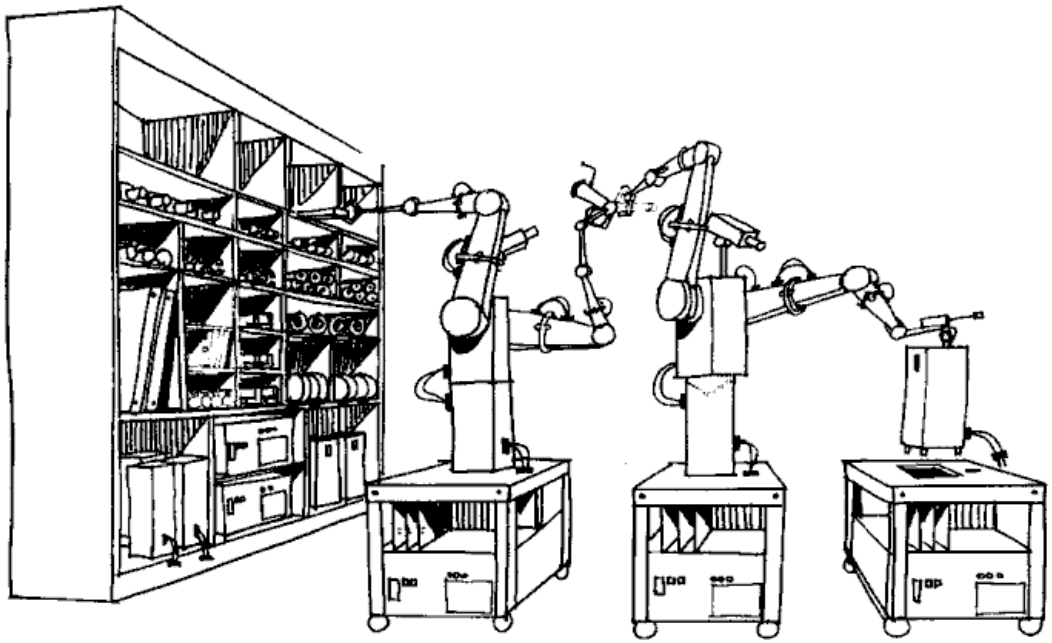
C++ тимлид и архитектор в LogicNow
IT-консультант в DPI.Solutions
В прошлом – разработчик в Microsoft в команде Hyper-V
Активист сообщества COMAQA.BY

www.dpi.solutions

Кодогенерация

Классическое разделение

1. Пассивная – разовая, с ручными правками
2. **Активная** – автоматическая, регулярная, без ручных правок

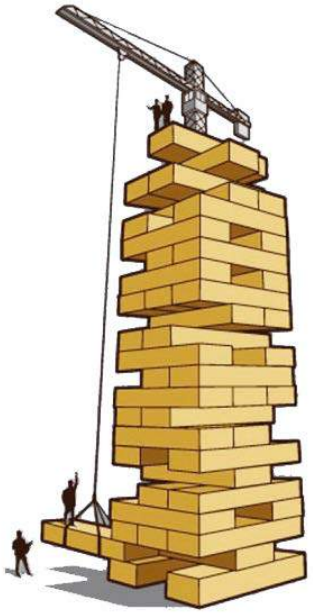


www.dpi.solutions

Частые релизы продукта

Вызовы

- быстрая реакция на изменение требований
- минимизация человеческих ошибок
- высокое покрытие тестами

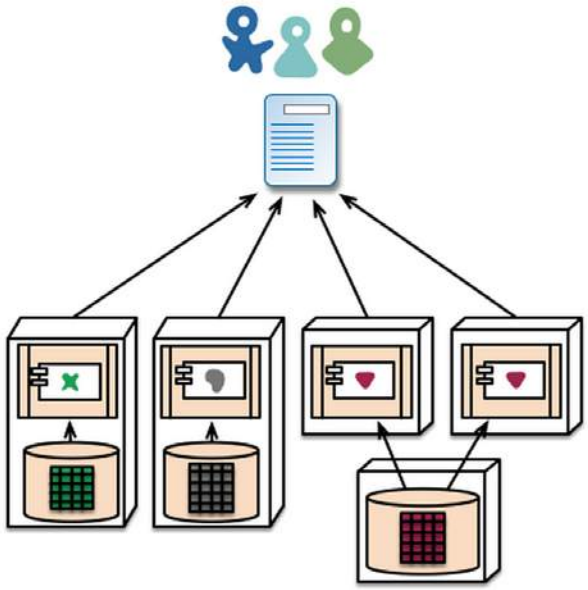


www.dpi.solutions

Microservice-архитектура

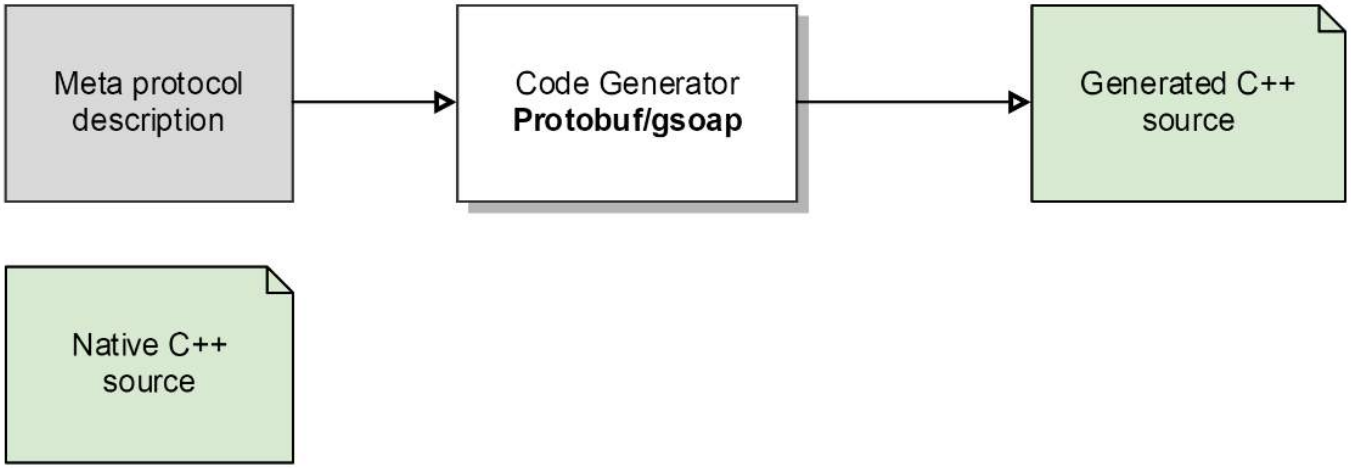
Рутинные задачи

- создание сетевых протоколов
- создание слоя хранения данных
- тесты, тесты, тесты



www.dpi.solutions

Как генерируют протоколы

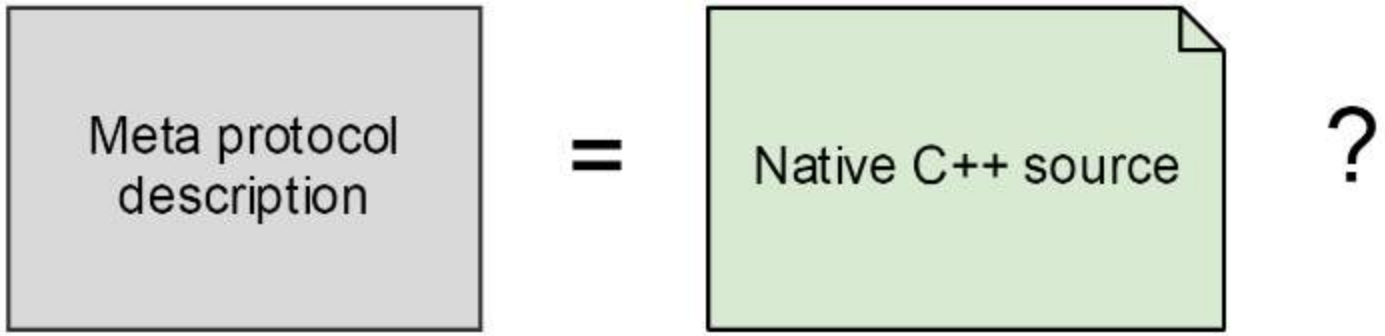


Минусы

- нет контроля над процессом генерации
- генерированные исходники “чужеродны” для проекта
- дублирование кода (“родные” сущности сосуществуют с генерированными)

www.dpi.solutions

Single Source of Truth?



www.dpi.solutions

Single Source of Truth!

Сущности

```
// CustomerInfo.h
struct CustomerInfo
{
    int Id;
    string Name;
    CustomerType Type;
};
```

Интерфейсы

```
// ICustomerManager.h
class ICustomerManager
{
        virtual int CreateCustomer(CustomerInfo const& customer) = 0;
        virtual CustomerInfo GetCustomer(int customerId) = 0;
        virtual void UpdateCustomer(CustomerInfo const& customer) = 0;
        virtual void DeleteCustomer(int customerId) = 0;

        virtual ~ICustomerManager() {}
};
```

Декларации на C++ и есть самодостаточное базовое мета-описание протокола

2015 CEE-SECR Software Engineering Conference in Russia

Реализация: ClangTool

ClangTool – парсер C++ деклараций (~500 строк кода)

На выходе: разобранный набор типов с переменными и методами

TypeA
[**Field1**: Type1, …, **FieldN**: TypeN]
[**Method1**: Params1, ... , **MethodM**: ParamsM]

А дальше ограничены только нашей фантазией

www.dpi.solutions

Реализация: TemplateTool

TemplateTool – кодогенератор на основе шаблонов

```
// StructSerialization.gen
void CppToJson(<%struct.Name%> const& native, Json::Value& json)
{
        json["typename"] = "<%struct.Name%>";
        <%foreach field in struct.Fields%>
        CppToJson(native.<%field.Name%>, "<%field.Name%>", json);
        <%end%>
}
```

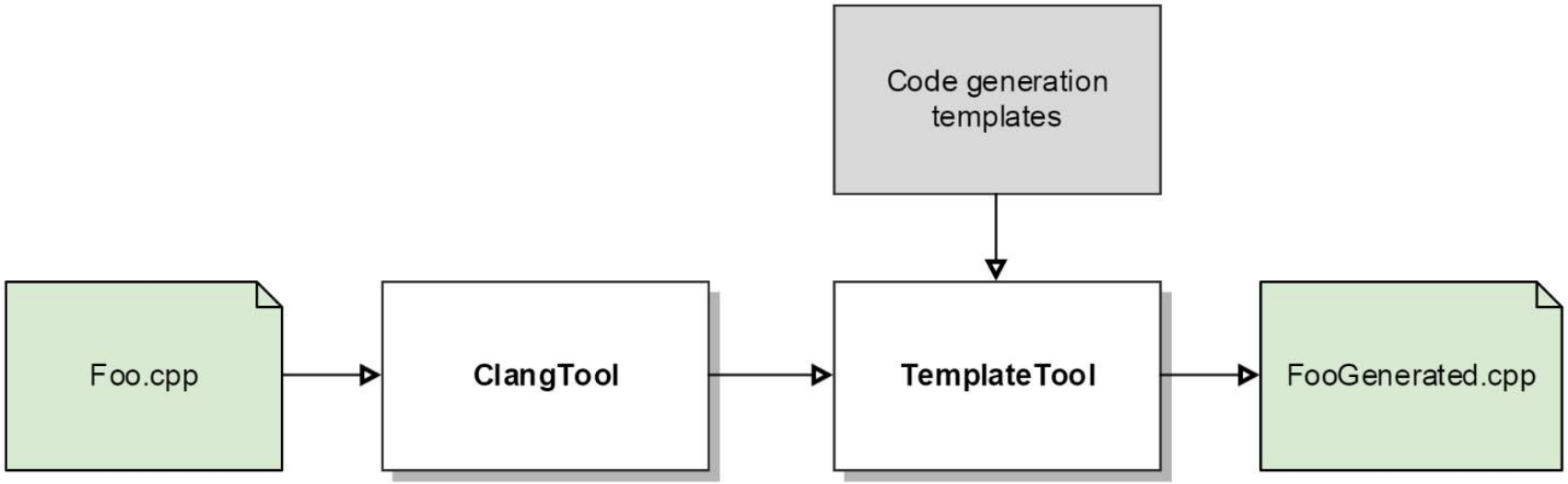
⇓

```
// CustomerInfoSerialization.cpp
void CppToJson(CustomerInfo const& native, Json::Value& json)
{
        json["typename"] = "CustomerInfo";

        CppToJson(native.Id, "Id", json);
        CppToJson(native.Name, "Name", json);
        CppToJson(native.Type, "Type", json);
}
```

www.dpi.solutions

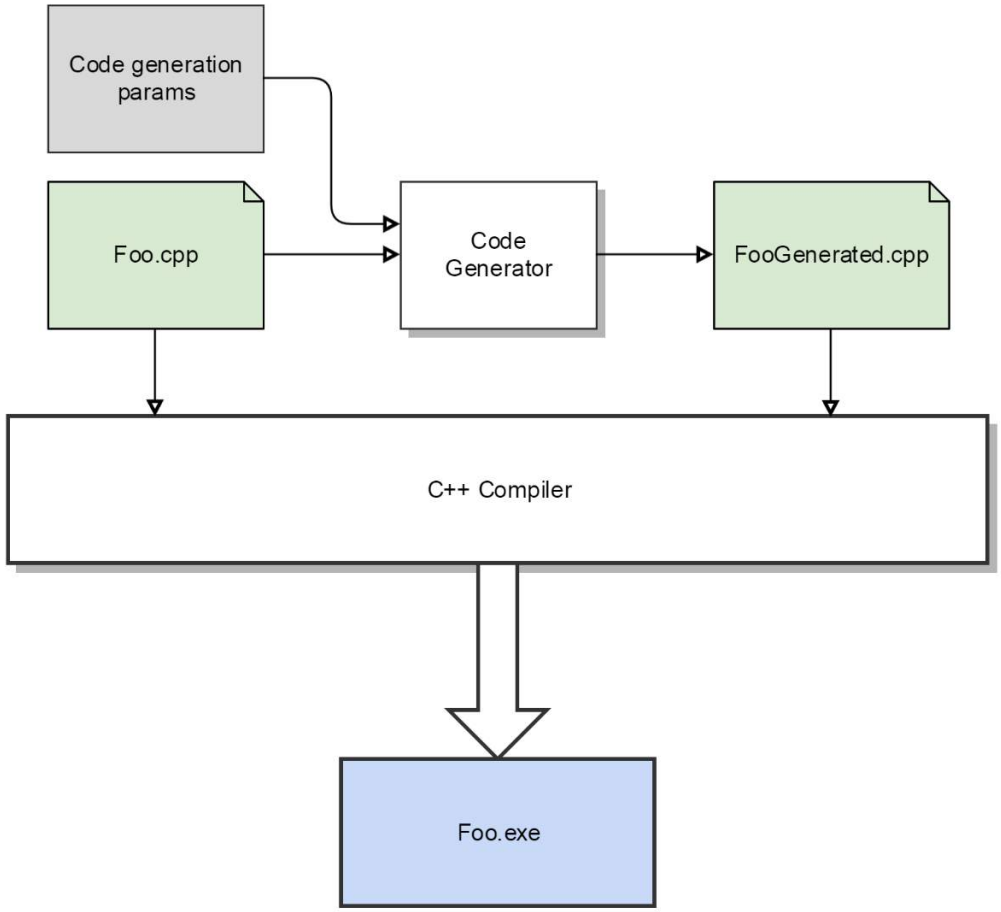
Схема кодогенерации



Шаблоны кодогенерации пишутся разово под класс задач

www.dpi.solutions

Общая схема компиляции



Code Generator = **ClangTool** + **TemplateTool**

Базы данных

```
// CustomerInfo.h
struct CustomerInfo
{
    int Id;
    CustomerType Type;
    string Name;
};
```

уже описанным способом

⇒

```
// CustomerInfo.ddl
CREATE TABLE CustomerInfo
(
    Id INT(4) NOT NULL,
    CustomerType  INT(4),
    Name CHAR(20)
)
```

Преобразуем C++ декларации в SQL (DDL)

www.dpi.solutions

Шаг дальше. Базы данных

Если недостаточно синтаксиса “по умолчанию”:

C++ 98/2003 (комментарии)

```
// CustomerInfo.h
struct CustomerInfo
{
    int Id;
    CustomerType Type; // FK: CustomerType.Id
    string Name;
};
```

C++ 11/14 (атрибуты)

```
// CustomerInfo.h
struct CustomerInfo
{
    int Id;
    [[FK: CustomerType.Id]]
    CustomerType Type;
    string Name;
};
```

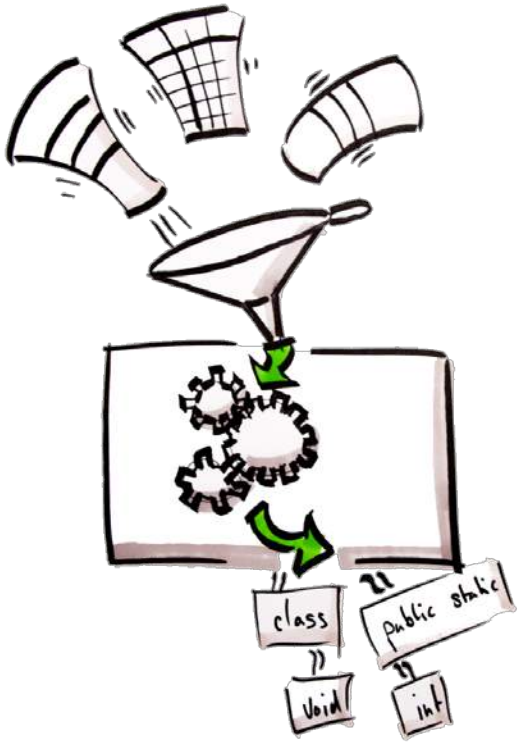
⇒

```
// CustomerInfo.ddl
CREATE TABLE CustomerInfo
(
    Id INT(4) NOT NULL,
    CustomerType  INT(4) NOT NULL
REFERENCES CustomerType(Id),
    Name CHAR(20),
    KEY CustomerType (CustomerType)
)
```

www.dpi.solutions

Тесты для протоколов и баз данных

Тоже можно сгенерировать!



www.dpi.solutions

Итого

Выгоды

- Устранение рутинной работы
- Минимизация человеческих ошибок
- Решение типового набора задач “за бесплатно”
- Более высокий уровень абстракции

Проблемы

- Версионность (реакция на изменения кода)
- Сопряжение с рукописным кодом
- Сложность отладки

www.dpi.solutions

Спасибо! Вопросы?

Антон Наумович
DPI.Solutions / LogicNow
naumovich@dpi.solutions

2015
CEE-SEC(R)
Software Engineering
Conference in Russia