DSL Tools for Model-Driven Development

Александр Гаврилов
alexga@microsoft.com
Microsoft

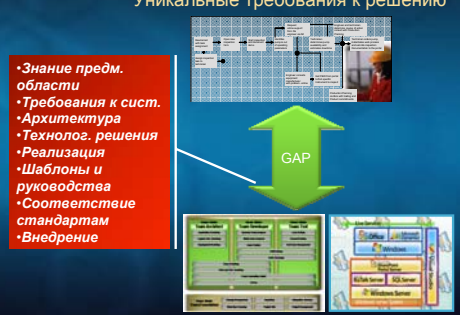
Елена Павлова
net.dot@gmail.com
МИФИ

Краткое содержание

- Что такое MDD и DSL
- Среды поддержки DSL
- Microsoft DSL Tools
- Демонстрация
- Заключение

Создание программной системы

Уникальные требования к решению



Платформы общего назначения

Проблемы сред разработки программных систем

- Сложность систем
- Особенности предметной области
- Формализация разработки
- Накопленный опыт
- Наглядность
- Гибкость
- Верифицируемость
- Реиспользуемость

- Время
- Стоимость
- Качество

Эволюция методов разработки

- 1960е – Сети Петри
- Начало 1970х – Структурное программирование (Боэм, Дейкстра)
- Середина 1970х – Структурное проектирование (Йордон-Константин), JSP (Джексон), ER-модели (Чен)
- Конец 1970х – Структурный анализ (ДеМарко, Гейн-Сарсон)
- Начало 1980х - Метрики
- Середина 1980х – Информационная инженерия (Мартин)
- Конец 1980х – Объектно-ориентированное проектирование (Буч)
- Начало 1990х – Объектно-ориентированный анализ (Коад-Йордон, Шлаер-Меллор, Румбах)
- Середина 1990х – Бизнес-моделирование, моделирование реинжиниринга бизнес-процессов (BPR Modeling), моделирование процессов документооборота (Work Flow Modeling)
- Конец 1990х – Стандартизация объектно-ориентированного подхода (UML), эталонное моделирование (reference Modelings)
- Начало 2000 – Моделирование при помощи компонент, языки моделирования, зависящие от предметной области

kontrac

Поддержка специфики ПО

- *Библиотеки подпрограмм* содержат подпрограммы, выполняющие взаимосвязанные задачи в рамках хорошо определённой предметной области. Примером могут служить дифференциальные уравнения, графика, пользовательские интерфейсы, базы данных. Создание библиотеки подпрограмм – классический метод накопления повторно используемых знаний, специфичных для предметной области.
- *Объектно-ориентированные и компонентно-ориентированные среды* продолжают идею библиотеки подпрограмм. Классические библиотеки имеют плоскую структуру, а приложения выполняют вызов библиотечных подпрограмм. В случае объектно-ориентированных сред часто происходит вызов специфичных для приложения методов из среды.
- **Язык, зависящий от предметной области (*domain-specific language* - DSL)** суть небольшой, как правило декларативный язык, чья выразительная мощность сосредоточена на конкретной предметной области. Во многих случаях программы на DSL транслируются в вызовы библиотечных подпрограмм, а DSL рассматривается как способ инкапсуляции элементов библиотеки.

Model-Driven Development (MDD)

- Процесс, «правильный по построению» (correct-by-construction) в отличие от традиционного процесса разработки вручную в стиле «построения путем коррекции» (construct-by-correction).
- *Управляемая моделями разработка* (Model-Driven Development - MDD) — это развивающаяся парадигма, решающая многочисленные проблемы композиции и интеграции крупномасштабных систем и опирающаяся при этом на достижения в области технологий разработки (в частности, на компонентное программное обеспечение промежуточного слоя).
  - *модельно-управляемая архитектура* (Model-Driven Architecture, MDA). Системы представляются с использованием языка моделирования общего назначения Unified Modeling Language и его конкретных профилей. Эти модели преобразуются в артефакты, выполняемые на разнообразных платформах.
  - ***модельно-интегрированные вычисления*** (Model-Integrated Computing, MIC). Для представления элементов системы и их связей используются предметно-ориентированные языки моделирования DSML, а также их преобразования в платформенно-зависимые артефакты.

Model-Integrated Computing (MIC)

- Модельно-интегрированые вычисления (Model-Integrated Computing, MIC) объединяют:
  - предметно-ориентированные языки моделирования DSML (Domain-Specific Modeling Language), в системах типов которых формализуется структура, поведения и требования приложения внутри соответствующей предметной области. В используемых метамоделях определяются связи между понятиями предметной области, точно специфицируется основная семантика и ограничения, ассоциируемые с этими понятиями.
  - трансформационные процессоры и генераторы, которые анализируют определённые аспекты моделей и синтезируют исходный код, входные данные для имитационного моделирования, XML-описания развертывания или альтернативные представления моделей. Возможность синтеза на основе моделей помогает поддерживать согласованность между реализациями и аналитической информацией о зафиксированных в модели требованиях к функциональным возможностям системы и ее качеству.

DS(M)Ls – языки моделей

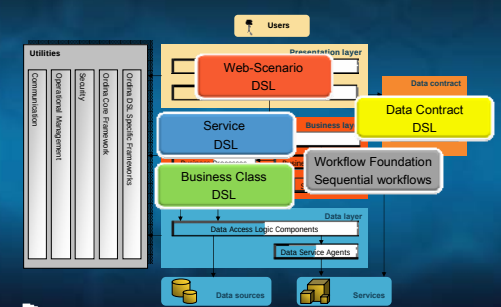
- Язык описания модели часто называют **метамодель**ю, следовательно, язык описания языка моделирования является мета-метамоделью. Мета-метамодели можно подразделить на две группы: языки, полученные видоизменением или наследованием от существующих языков, и языки, специально разработанные как мета-метамодели.
- Производные мета-метамодели включают диаграммы "сущность-связь" (ERD), формальные языки, РБНФ (EBNF), языки онтологий, XML-схемы и др. Преимущество этих языков состоит в том, что они построены на основе известных и стандартизованных исходных языков.
- Характер моделирования, зависящего от предметной области, способствует появлению новых языков для конкретных задач. Таким образом, создаются новые языки, разработанные как мета-метамодели.
- Язык программирования, зависящий от предметной области (**DSL**) есть язык программирования, спроектированный для определённого круга задач, в противоположность языкам программирования общего назначения (GPL) таким как PL/1, ADA или языкам моделирования общего назначения таким как UML. Примерами DSL могут служить матрицы обработки таблиц, YACC - язык для синтаксического анализа и компиляции, Csound- язык создания аудиофайлов и GraphViz– язык описания и визуализации ор-графов.

Пример: Графические языки

Важны
- Блоки
- Связи, стрелки, стили
- Пометки
- Вложенность
- Расположение
- Маршрутизация

Используются
- Для понимания и описания требований
- Для понимания и описания архитектуры
- При декомпозиции решения на модули
- При применении шаблонов проектирования
- Для описания и визуализации существующих систем

Архитектура – место DSL



11

Методология разработки DSL

Разработка языка, зависящего от предметной области, как правило включает следующие шаги:

- **Анализ**
  - (1) Определение предметной области.
  - (2) Сбор всех релевантных знаний предметной области.
  - (3) Построение системы знаний в виде нескольких семантических нотаций и операций над ними.
  - (4) Проектирование DSL, лаконично описывающего приложения в предметной области.
- **Реализация**
  - (5) Создание библиотеки, реализующей семантические нотации.
  - (6) Проектирование и реализация компилятора, транслирующего DSL-программы в последовательность библиотечных вызовов.
- **Применение**
  - (7) Написание и компиляция необходимых DSL-программ.

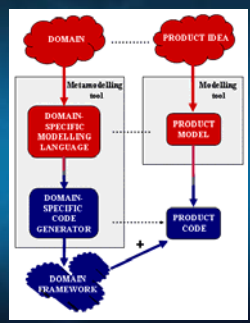
Преимущества и недостатки подхода

- Преимущества DSL:
  - DSL позволяет выразить решение в терминах предметной области на соответствующем уровне абстракций. Следовательно, специалисты в данной предметной области могут разбирать, верифицировать, изменять и разрабатывать DSL-программы.
  - DSL-программы лаконичные, во многом самодокументирующиеся и могут повторно использоваться для различных целей.
  - DSL повышает эффективность, надёжность, переносимость и качество сопровождения.
  - DSL содержит знания о предметной области, обеспечивая таким образом возможность их хранения и повторного использования.
  - DSL позволяет проводить валидацию и оптимизацию на уровне абстракций, соответствующем предметной области.
  - DSL улучшает тестируемость.
- Недостатки применения DSL:
  - Стоимость проектирования, реализации и сопровождения DSL.
  - Стоимость обучения пользователей DSL.
  - Ограниченная доступность различных DSL.
  - Сложность определения области действия DSL.
  - Сложность сохранения равновесия между конструкциями, специфичными для предметной области, и конструкциями языков программирования общего назначения.
  - Возможное снижение эффективности по сравнению с программным обеспечением, целиком написанном на языке программирования общего назначения.

Среды поддержки DSML

- Как правило архитектура среды разработки DSL состоит из трёх частей:
- 1. грамматическая система языка, определяющая синтаксис и множество правильных предложений;
- 2. библиотека компонент, описанных на результирующем языке (DSL);
- 3. преобразования, определяющие семантику DSL

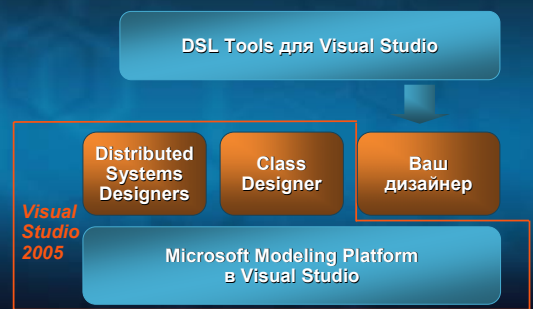
На данный момент предметно-ориентированное моделирование чаще всего происходит при помощи предметно-ориентированных сред, промышленных (MetaEdit+) или научных (GME). В связи с ростом популярности предметно-ориентированного моделирования соответствующие среды стали встраивать в уже существующие IDE, (например Eclipse Modeling Project (EMP) и Microsoft's DSL Tools для Software Factories).



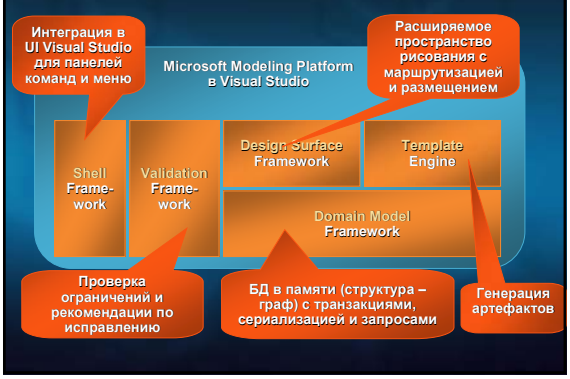
Microsoft Domain Specific Language (DSL) Tools для Visual Studio 2005

- Позволяют создать выполняемый дизайнер для языка предметной области
  - Полноценный инструмент (полнофункциональный визуальный редактор, верификация модели, сохранение в XML, генерация текста и т.д.)
  - Designer SDK (для создания новых дизайнеров)
    - Model Data Access
    - Model Validation
    - UI and Designer Behavior
    - Custom XML Serialization
  - Richer design experience for Domain Models, Notation and Mapping
  - Multiple Views and Multiple Models
  - Designer Deployment Support
- Текущие ограничения реализации:
  - Языковые конструкции – ExtER
  - Трансформационные преобразования – алгоритмы Маркова
  - Ограничения на модели – логика I порядка

Архитектура DSL Tools

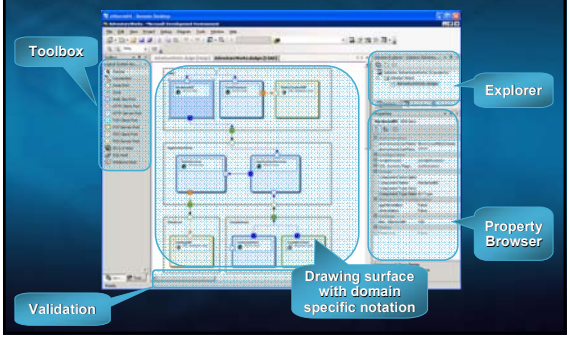


Microsoft Modeling Platform

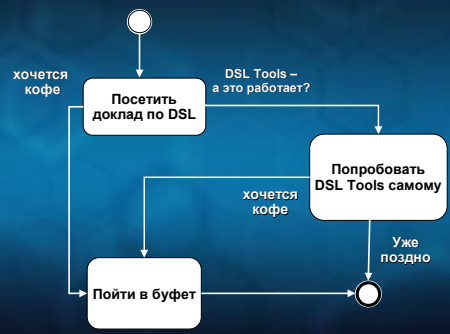


Создаем дизайнер

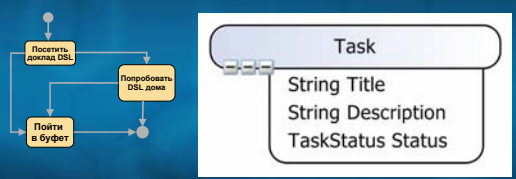
Создание дизайнера для Visual Studio



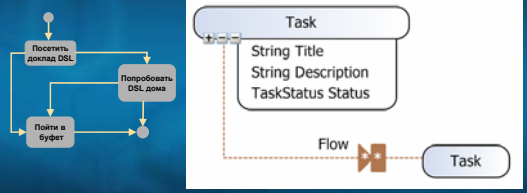
Простой язык workflow



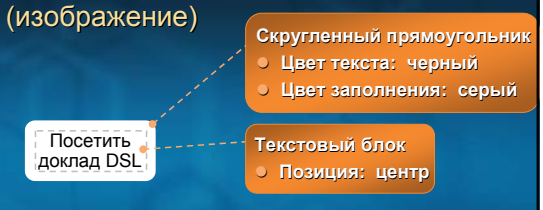
Описываем модель
Класс задачи



Описываем модель
Связи



Описываем класс
(изображение)

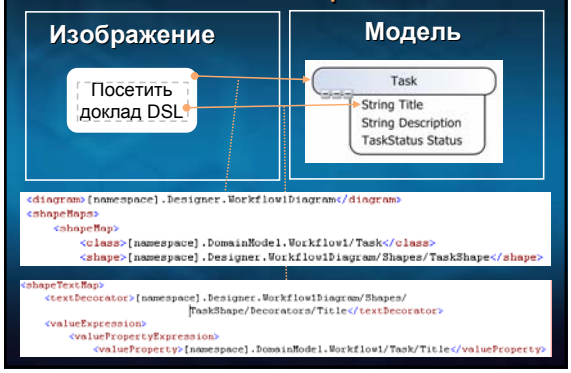



```
<geometryShape name="TaskShape" initialWidth="1.5" initialHeight="0.6"
               geometry="RoundedRectangle">
    <decorators>
        <shapeText name="Title" position="Center" defaultTextId="Title"/>
    </decorators>
    <fillColor color="gray" variability="User"/>
    <outlineColor color="black" variability="Fixed"/>
```

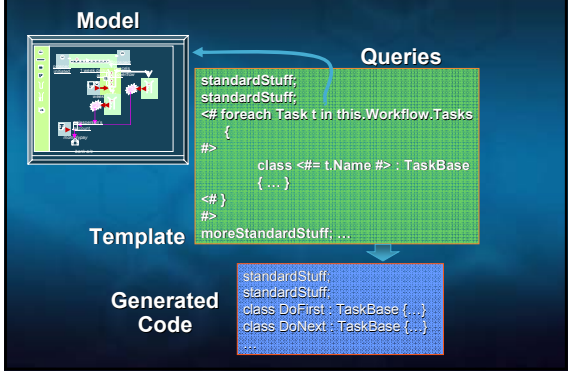
Описываем связь
(изображение)


```
<connector name="FlowLink">
    <color variability="User" color="black" />
    <dashStyle variability="User" dashStyle="solid"/>
    <decorators>
        <connectorText name="Label" position="SourceTop"
```

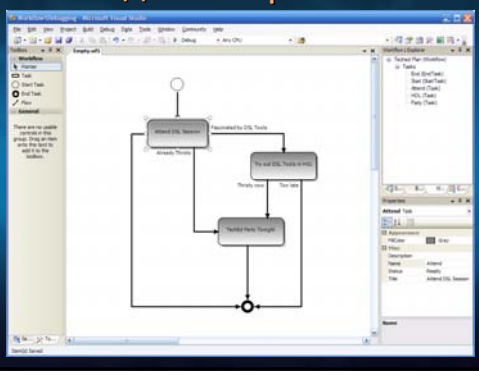
Описываем отображение



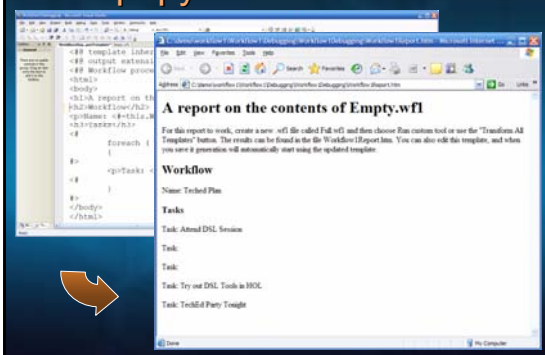
Генерация кода (Templates)



Новый дизайнер



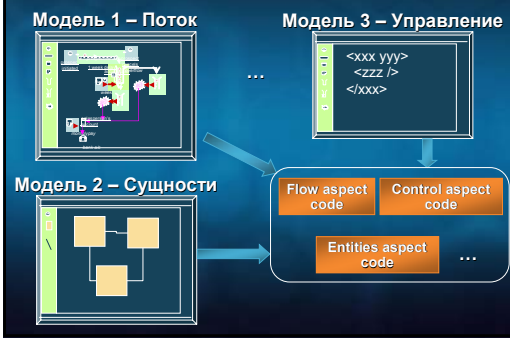
Генерируем отчет



demo

Схема освещения

Использование DSL для индустриализации разработки – Software Factories



Источники (MDD, DSL)

- Douglas Schmidt, Model-Driven Engineering ,Computer (IEEE Computer Society)
- Arie van Deursen, Paul Klint, Joost Visser, Domain-Specific Languages: An Annotated Bibliography
- Don Batory, The Road to Utopia: A Future for Generative Programming (Domain-Specific Program Generation)
- Charles Consel, From a Program Family to a Domain-Specific Language
- Charles Consel, Laurent Reveillere, A DSL Paradigm for Domains of Services: A Study of Communication Services
- Jean Bezevin, Expert's Voice on the Unification Power of Models (Software System Models)
- John D. Poole, Model-Driven Architecture: Vision, Standards And Emerging Technologies (ECOOP 2001)
- Kontrac, From Coding to Modeling: Past, Present, Future Metacase Domain-specific modelling (Application Development Advisor)

Источники (Microsoft DSL Tools)

Домашняя страница DSL Tools
http://lab.msdn.microsoft.com/teamsystem/workshop/dsltools/

Форум по DSL Tools
http://forums.microsoft.com/msdn/ShowForum.aspx?ForumID=61

Фабрики ПО (Software Factories)
http://lab.msdn.microsoft.com/teamsystem/workshop/sf/

Вопросы?

Александр Гаврилов
alexga@microsoft.com

Елена Павлова
net.dot@gmail.com