Методы Программной Инженерии в Индустрии Компьютерных Игр

Елена Павлова
Московский инженерно-физический институт (государственный университет)
Elena.pav@gmail.com

Александр Гаврилов
ООО «Майкрософт Рус»
Alexander.Gavrilov@microsoft.com

Применение предметно-ориентированных языков для разработки компьютерных игр

Содержание

- Обзор индустрии компьютерных игр
- Проблемы, стоящие перед разработчиками игр
- Проблема разработки правил игры в классе пошаговых стратегий
- Обзор подходов к разработке правил игры
- Иллюстрация предлагаемого в настоящей работе решения

Компьютерная игра – игра, существующая в виде программы, выполняемой на компьютере.

Определение не включает игры, выполняемые на мобильных устройствах, игровых консолях (“приставках”), игровых автоматах.

Доходы рынка видеоигр

По данным Forbes, CNews, Games Industry

Компьютерная игра – разновидность программного обеспечения

Специфика индустрии разработки компьютерных игр

по сравнению с индустрией разработки программного обеспечения

- Развивалась как часть индустрии развлечений, а не индустрии разработки программного обеспечения
- Особенно сильна творческая составляющая
- Недостаточно специализированных средств разработки
- Проблема качества
- Ежегодно в мире создается от 3 до 5 тысяч компьютерных игр, из них несколько десятков становятся программными продуктами и появляются в продаже

Специфика требований к компьютерным играм

- Заказчик – издатель. Единственная цель заказчика – получение максимальной выгоды от продаж
- Широкая аудитория - дополнительные трудности при сборе требований
- Творческий и инициативный аспекты часто усложняют сбор требований
- Игры очень быстро устаревает морально, поэтому требования к срокам разработки игр выше
- Одно из основных - требование минимизация времени обучения игре
- Часто высокие требования к аппаратному обеспечению
- Требование поддержки устройств, которые редко используются другими классами программ (например, различные манипуляторы, рули, педали, сенсорные ковры)
- Высокие требования производительности к ряду компонент игры, например, к коду графических подсистем.

Специфика проектирования компьютерных игр

- Сценарии игр ближе к сценариям кинофильмов, чем к UseCase в смысле [Gamma, Clockburn]
- Сценарий игры и концепция создаётся нетехническим специалистом
- Значительную часть проекта составляют работы дизайнеров, художников, музыкантов, специалистов по видео и трёхмерной графике. Процессы и методы для этой части проекта существенно отличаются от принятых в индустрии разработки программного обеспечения.
- Циклическая разработка для художественной части игрового проекта считается слишком затратной. Модель работы напоминает водопадную, что отражается на общей модели разработки и качестве игры
- Существует общепринятое деление игры на высокоуровневые компоненты
- Некоторые аспекты проектирования и реализации игр достаточно хорошо проработаны. Существуют в виде игровых движков (компонент). Несколько таких компонент используется при разработке практически каждой игры, что накладывает значительные ограничения на проектирование и реализацию игры.

Специфика реализации компьютерных игр

- практически в каждом проекте используется сложная с точки зрения индустрии программного обеспечения графика. При этом доля проектов, использующих трёхмерную графику, продолжает расти.
- значительная часть игр предполагает взаимодействие двух и более (до нескольких миллионов) игроков по сети в режиме «мягкого» реального времени.
- значительная часть игр использует элементы искусственного интеллекта для моделирования поведения персонажей в игре.
- в играх часто применяются системы захвата движения и сопутствующие программные модули

«Узкие места» разработки компьютерных игр

- Производительность графических подсистем
- Управление данными игры
- Качество игры
- Скорость разработки игры
  - Автоматизация разработки
  - Повторное использование
  - Генерация кода и других артефактов

Иллюстрация применения методов программной инженерии в индустрии компьютерных игр

Разработка правил игры

в классе пошаговых стратегий

Понятие правила игры

Правила игры включают описание

- сущностей предметной области конкретной игры
- правил допустимого взаимодействия сущностей друг с другом
- главной цели игры и второстепенных целей
- условий выигрыша
- правила определения состояния игрового мира для каждого игрока, в том числе для момента начала игры

Пример правила игры

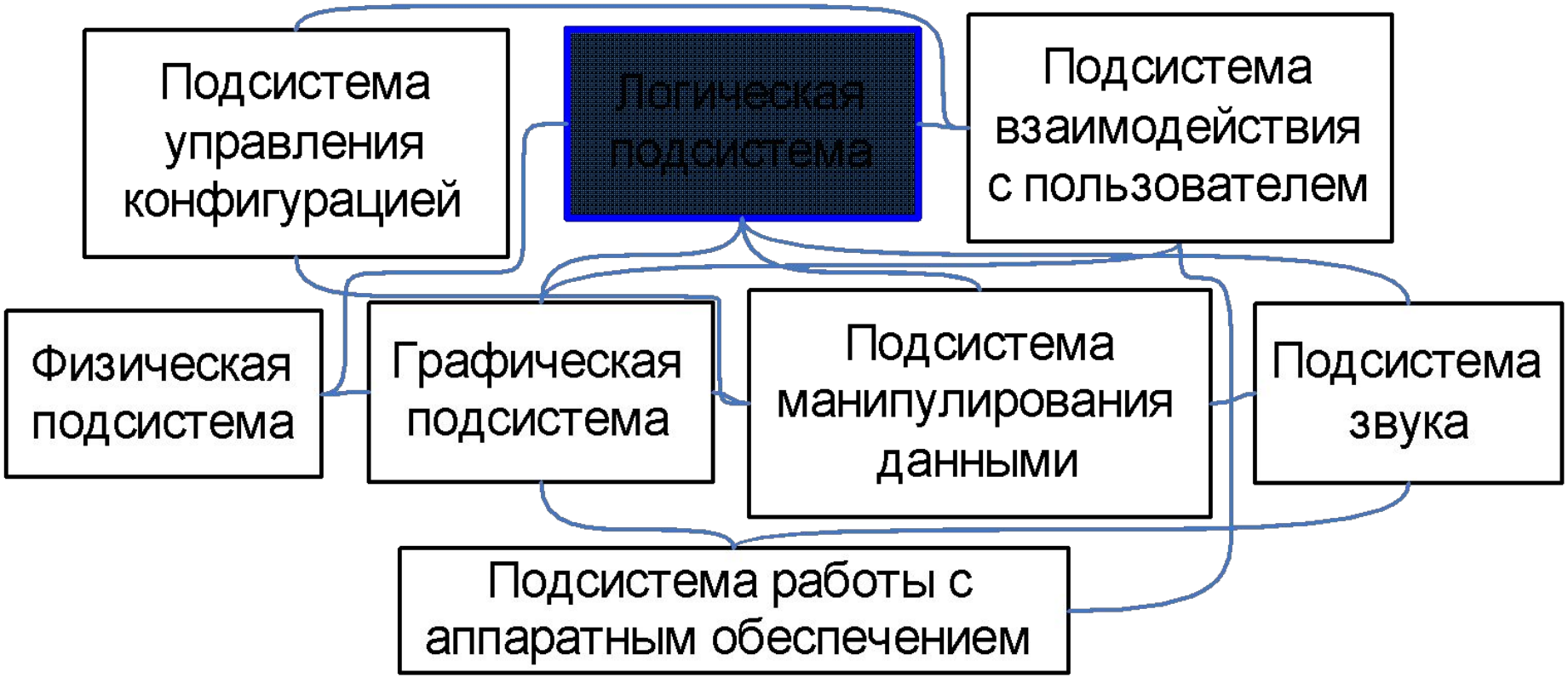
Отряд, сделавший шаг на клетку, где лежат сапоги-скороходы, может взять их при условии, что

- клетка не занята отрядом соперника
- Общий опыт отряда больше опыта, необходимого для использования сапог-скороходов
- Игрок успешно решил загадку, связанную с сапогами-скороходами

Когда отряд заберет сапоги-скороходы должно быть верно следующее:

- Сапоги-скороходы должны не принадлежать клетке
- Общая скорость отряда должна удвоиться

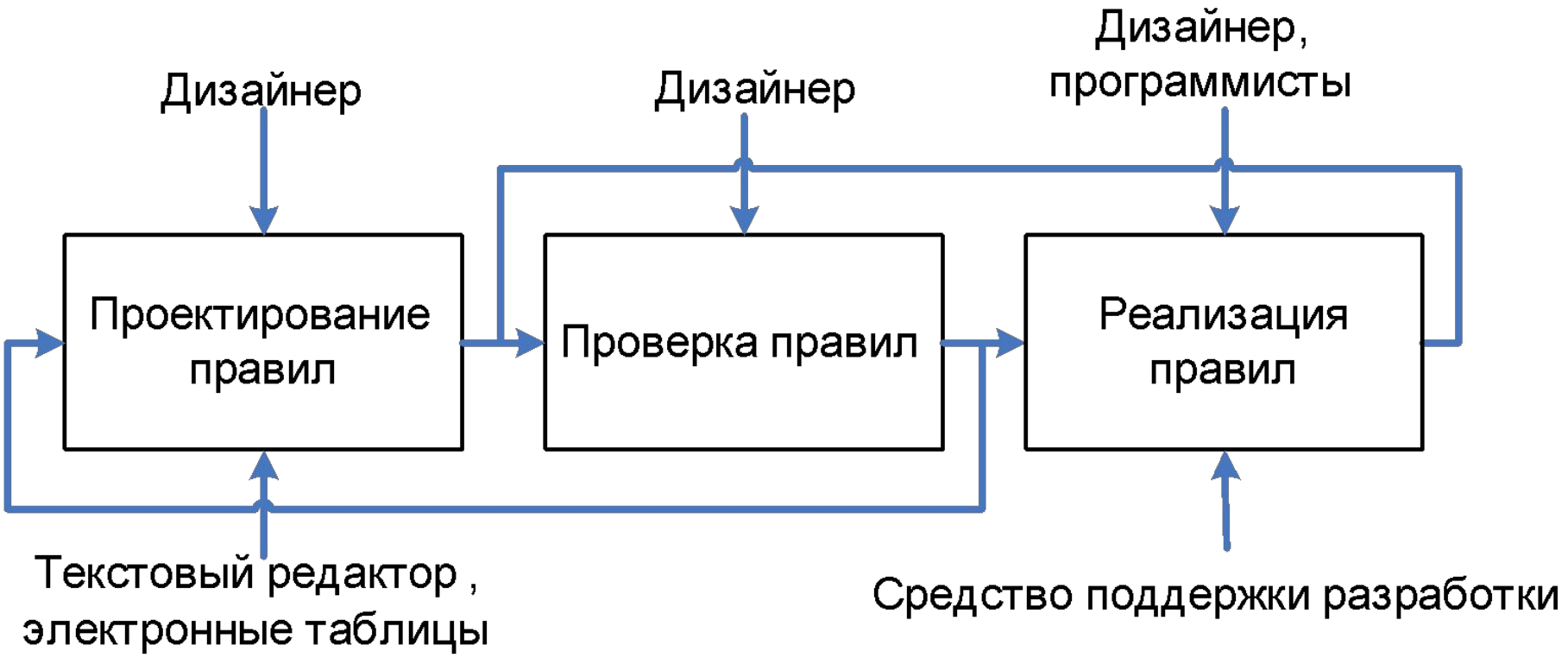
Правила – основа логической подсистемы игры



Логическая подсистема определяет сюжет игры; вымышленный мир, в котором будет действовать игрок; реакцию игры на действия игрока

Разработка правил

Разработка правил – часть процесса разработки игры



Недостатки текущего процесса разработки правил

- Значительные затраты времени и других ресурсов
- Низкая эффективность
  - Низкая степень повторного использования
  - Дублирование
  - Проверка правил часто выполняется вручную
- В текущем виде плохо поддаётся автоматизации
  - Результаты этапа проектирования и проверки правил часто объёмны неформальны и слабо структурированы
  - Не существует общепринятого алгоритма проверки правил

Постановка задачи

Разработка способа представления правил, который позволит

- автоматизировать процесс разработки правил
- учесть специфику жанра
- применить формальные методы балансировки и верификации к правилам игры.

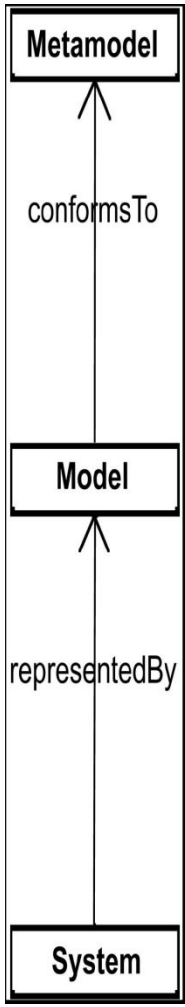
Предлагаемый подход к решению: применение графических моделей и предметно-ориентированных языков для представления правил игры

Model-Driven Development (MDD)

- Процесс, «правильный по построению» (correct-by-construction)— это развивающаяся парадигма, решающая проблемы композиции и интеграции систем и опирающаяся 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 плаформенно-зависимые артефакты.

DSLs – языки моделей

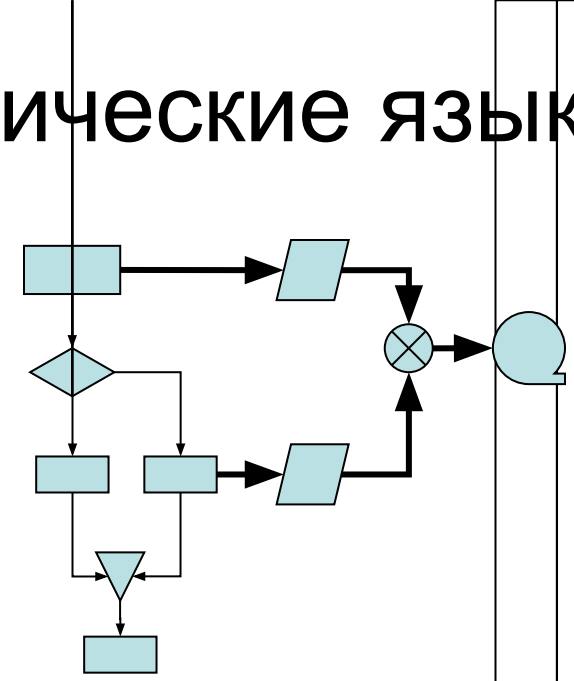
- .
- Характер моделирования, зависящего от предметной области, способствует появлению новых языков для конкретных задач. Таким образом, создаются новые языки, разработанные как мета-метамодели.
- Язык программирования, зависящий от предметной области (**DSL**) есть язык программирования, спроектированный 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.

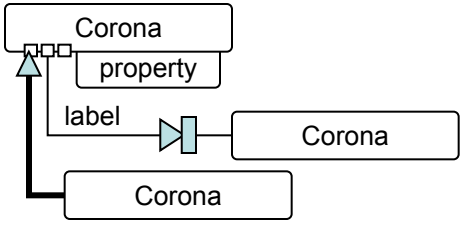


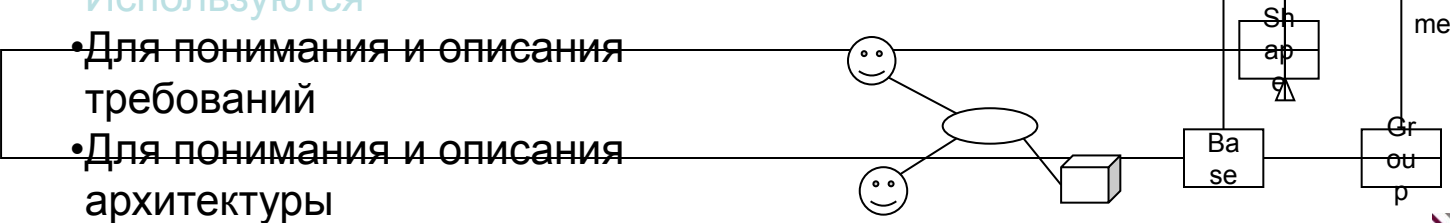
Пример: Графические языки

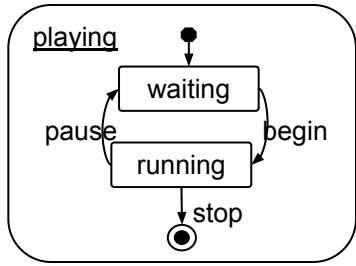
Важны

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



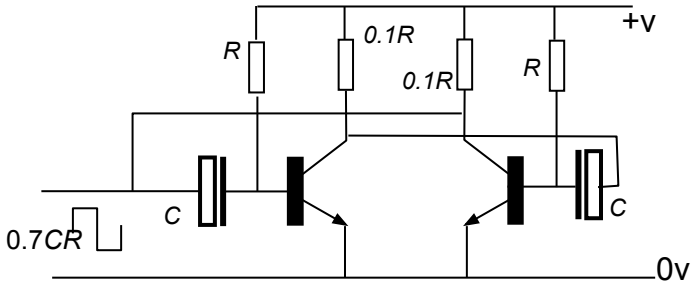






Используются

- 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



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 абстракции..
  - 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 абстракции, 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.

Вывод – при нашем подходе необходима разработка предметно-ориентированного языка для описания правил игр в классе пошаговых стратегий

Методика разработки правил (использования DSL)

Шаг 1: Анализ предметной области

Шаг 2: Разработка предметно-ориентированного языка (DSL) для описания правил игры

Шаг 3: Разработка графического предметно-ориентированного языка моделирования правил

Шаг 4: Разработка алгоритмов проверки моделей DSL

Шаг 5: Разработка алгоритмов генерации кода и других артефактов по моделям DSL

Шаг 6: Разработка инструментального средства поддержки процессов разработки, проверки моделей DSL и генерации кода

Шаг 7: Тестирование, внедрение, апробация

При этом на каждом этапе могут использоваться сторонние ресурсы в виде алгоритмов, фрагментов кода, готовых компонент, систем поддержки разработки и т.д.

Что даёт формализация?

Формальное представление правил упрощает

- Автоматизацию процесса проектирования правил
- Автоматизацию проверки правил
- Генерацию кода и данных на основе правил

Демонстрация: Язык представления правил

Источники информации

[1] Rollings A., Morris D. Game Architecture and Design. A New Edition.– Indianapolis: New Riders Publishing, 2004

[2] Androvich M. Forbes: Europe is least mature videogame market. – http://www.gamesindustry.biz/articles/forbes-europe-is-least-mature-videogame-market, 2008

[3] Путинцев Т. Игровая индустрия растёт. – http://www.dtf.ru/news/read.php?id=31442, 2003

[4] В Китае резко возросло число игроков. – http://www.lenta.ru/news/2007/05/07/china/

[5] Бондарева Т.В., Грызлова О.В., Евтюхин Н.В. Компьютерные деловые игры как инновационное средство обучения /Телекоммуникации и информатизация образования, (2001), 1 (январь), 58-69

[6] Плюммер Дж. Гибкая и масштабируемая архитектура для компьютерных игр. – http://www.dtf.ru/articles/read.php?id=40757, 2004

[7] Материалы конференции разработчиков компьютерных игр. КРИ 2007, – http://kriconf.ru/2007/index.php?type=info&doc=speech_records, 2007

[8] Wihlidal G. Game Engine Toolset Development.– Boston, MA: Thomson Course Technology PTR, 2006

[9] Meigs T. Ultimate Game Design: Building Game Worlds. NY: McGraw-Hill/Osborne, 2003

[10] Bates B. Game Design. Second Edition. – Boston, MA: Thomson Course Technology PTR, 2004

[11] Консалтинговая группа MD. – http://www.md-consulting.ru/

[12] Gruhl R. XNA Game Studio Express, an Overview. Proceedings of GDC 2007 – https://www.cmpevents.com/sessions/GD/S4440i1.ppt , 2007

[13] Flynt J. P., Salem O. Software Engineering for Game Developers. – Boston, MA: Thomson Course Technology, 2004

[14] Rucker R. Software Engineering and Computer Games.– London: Addison Wesley, 2003

[15] Yang H-C. A General Framework for Automatically Creating Games for Learning. Proceedings of ICALT'05. – IEEE, 2005, p. 28-29

[16] McNaughton M., Cutumisu M., Szafron D., Schaeffer J., Redford J., Parker D. Script-Ease: Generating Script-Code for Computer Role-Playing Games. Proceedings of ASE'04. – IEEE, 2004, p.386-387

[17] Meng L.S., Prakash E. C., Loh P. K. K. Design and Development of a Peer-to-Peer Online Multiplayer Game Using DirectX and C#. – IEEE, 2004, p. 278-281

[18] Common Questions about Blizzard – http://www.blizzard.com/us/inblizz/genfaq.html, 2008

[19] Guide to the Software Engineering Body of Knowledge. 2004 Version. SWEBOK. – Los Alamitos, California: IEEE, 2004

[20] Гаврилов А.В., Павлова Е. А. Формализация проектирования сложных информационных систем на основе анализа функциональных интерфейсов.

Информационные технологии №9, 2008. – М.: Новые технологии, 2008, стр. 9-15

Выводы

- Анализ индустрии показал актуальность задачи повышения качества игр за счёт применения методов программной инженерии в разработке игр.
- Предложенный подход на основании предметно-ориентированного языка дает возможность разработать инструментальное средство, позволяющее более качественно проектировать правила игры.
  На примере использования языка была продемонстрирована эффективность методов программной инженерии для решения задач разработки компьютерных игр.

Елена Павлова
Elena.pav@gmail.com

Александр Гаврилов
Alexander.Gavrilov@microsoft.com