ПРОБЛЕМЫ ОПТИМАЛЬНОГО ПРОЕКТИРОВАНИЯ В РАЗРАБОТКЕ ИГРОВЫХ ПРИЛОЖЕНИЙ

THE PROBLEM OF OPTIMAL DESIGN IN THE DEVELOPMENT OF GAMING APPLICATIONS

V. Kolychev
A. Petrov

Summary. The problems of design of game applications are considered and ways of their elimination are revealed. The concepts of correct solution and optimal solution are considered. The problems in the development of computer games are revealed. The risks of creating game applications are considered. The requirements that contribute to the elimination of significant problems that interfere with the proper design of gaming applications at the stage of their development.

Keywords: Computer games, social networks, development, industry, design.

Колычев Виктор Сергеевич
Аспирант, Российский технологический университет (МИРЭА)
svkolychev@yandex.ru
Петров Андрей Борисович
Д.т.н., Российский технологический университет (МИРЭА)
petrov@mirea.ru

Аннотация. Рассмотрены проблемы проектирования игровых приложений и выявлены способы их устранения. Рассмотрены понятия правильного решения и оптимального решения. Выявлены проблемы в разработке компьютерных игр. Рассмотрены риски при создания игровых приложений. Предложены требования, которые способствуют устранению значимых проблем, которые мешаю правильному конструктированию игровых приложений на этапе их разработки.

Ключевые слова: Компьютерные игры, социальные сети, разработка, индустрия, проектирование.

На разных этапах проектирования (особенно часто на начальных этапах) перед разработчиком стоит задача выбора наилучшего варианта из множества приемлемых проектных решений, отвечающих требованиям.[2, с. 25]

Неизбежная плата за попытку получить решение в условиях неполной информации об объекте проектирования является возможностью неверных решений. Поэтому в такой ситуации лицо, принимающее решение (ЛНР), должно разработать стратегию принятия решения, которая, не исключая возможности принятия неправильных решений, минимизирует негативные последствия. Чтобы уменьшить неопределенность, ЛНР может провести эксперимент, но это дорого и отнимает много времени. Поэтому ЛНР должна определиться с формой, временем, уровнем эксперимента.

Само решение, это компромисс. При принятии решения необходимо взвешивать ценностные суждения, которые включают в себя учет многих факторов, в том числе экономических, технических, научных, эргономических, социальных и других.

Понятие «правильного» решения означает выбор альтернативы из числа возможных вариантов, при которой общая стоимость будет оптимизирована с учетом всех различных факторов. Процесс принятия решений в оптимальном проектировании характеризуют следующие основные черты: наличие целей (показателей) оптимальности, альтернативных вариантов проектируемого объекта и учет существенных факторов при проектировании.

Понятие «оптимальное решение» при проектировании имеет вполне определенное толкование, лучшее в том или ином смысле проектное решение, допускаемое обстоятельствами. В подавляющем большинстве случаев одна и та же проектная задача может быть решена несколькими способами, что приводит не только к различным выходным характеристикам, но и к классам программ.

В современных условиях вопрос качества программного обеспечения, поставляемого на рынок, становится все более актуальным — с ростом числа девелоперских компаний, а также независимых команд программистов, готовых предоставить потребителям готовый продукт, возрастает интерес к скорейшему завершению реализации коммерческих решений. Зачастую в погоне за прибылью ключевые проблемы разработки программного обеспечения уходят на второй план — в борьбе за качественные программные решения преобладают сроки. Отметим основные факторы, которые оказывают негативное влияние на качество выпускаемых на рынке программ.

Индустрия разработки игр очень высоко рисковая. Некоторые риски можно снизить относительно легко, другие наоборот, очень трудно. Рисков много и они разноплановые.[1, с. 35]

Для кого-то очевидны управленческие проблемы и как их решать, для других — технические, для третьих — экономические. Итак, риски.

Из чего состоит игра? Минимум из трех основных компонентов: техническая реализация, графическое изображение и геймдизайн. Для больших игр (например, MMO) техническая часть делится на клиента (игру, то, что видят игроки и с чем взаимодействуют) и сервер (то, где происходит сама игра, обсчеты физики, путей, логики и многого другого). Для социальных сетей не менее важна виртуальность (вирусное распространение), без которого игра не будет прибыльной. Если некачественный игровой дизайн, то в игру не будут играть. Если не качественная графика, то игру даже смотреть не будут. И так далее, для каждой платформы могут быть свои особенности, которые надо знать. Таким образом, каждая из основных компонентов игры должна быть не ниже определенного уровня, который ожидают получить игроки. По мере развития игровой индустрии, планка для каждой компоненты поднимается все выше, что, в свою очередь, поднимает порог вхождения в индустрию для непрофильных инвесторов или молодых разработчиков.[4, с 47]

Этот риск вытекает из предыдущего. Для того, чтобы каждое направление было бы на уровне, необходимы ведущие специалисты (лиды). Стандартный набор — лид-артист (арт директор), лид-программист (ведущий программист или архитектор), продюсер (часто совмещает должность ведущего геймдизайнера). Каждый из этих сотрудников должен обладать не только высочайшей квалификацией, но и понимать игры и чувствовать на уровне интуиции, что для игры хорошо, а что — плохо. Найти таких ключевых людей сложно и, по большей части, они все уже разобраны и замотивированы окладами, на 20–50% выше чем по рынку. Например, грамотного продюсера найти в Москве сейчас меньше, чем на 170–200 тысяч будет наверное уже невозможно. Для других городов России все попроще, цены могут быть и вдвое ниже.

Если замену художнику можно найти (простите, господа артисты!) относительно легко только потому, что его работа видна сразу, то если исчезнет ваш ведущий программист, что будет? Если пропадет ваш ведущий геймдизайнер (а часто он вообще один на проекте), то кто знает, куда дальше развивать проект и как? Ведь геймдизайнер ставит целью далекую перспективу и часто (честно — почти всегда) все будущие механики не описаны. В любом случае, смена ведущего специалиста на игровом проекте это всегда переделки. Если очень сильно повезет, то переделывать надо будет немного (редко). В обычной же ситуации переработке может подвергнуться треть всего проекта, а в наихудших случаях — все направление (арт, код, геймдизайн) будет переделано с нуля.

Этот риск снижается мотивационными механизмами и тем, что на одном направлении работает не по одному человеку, а минимум по двое, которые могут делать работу друг друга или, в крайнем случае, продолжить ее в случае ухода сотрудника до тех пор, пока не будет найдена замена. Это — очевидная вещь, которую, тем не менее, мало кто практикует.

Достаточно большие команды разработчиков, планирующих создание нового программного обеспечения, как правило, совершают первую ошибку на этапе формирования технического задания. Вопреки проверенной технологии качественной разработки программного обеспечения, заказчики не редко ограничивают сроки реализации проекта, заставляя команду разработчиков исключить из перечня задач процесс отладки написанного программистами кода. Вместо оптимизации каждого компонента программного продукта задача «отлова багов» возлагается на тестировщиков, которые берутся за работу на последнем этапе разработки программного обеспечения, когда все модули программы уже собраны воедино.[3, с. 87]

Чуть менее существенной ошибкой, приводящей к проблемам в разработке программного обеспечения, является игнорирование очень необходимой процедуры анализа требований к будущей программе, сформированной заказчиком и исполнителем. Расплывчатое понимание целей, преследуемых заказчиком, а также несогласованность бизнес-деталей, которые должны быть реализованы в продукте, приводит к поставке некачественных программ со значительным количеством дефектов. Эта проблема особенно остро стоит в сфере разработки мобильных приложений, где ключевым фактором для клиентов является ранний запуск решения на рынок с целью заработать деньги от продажи программы потребителям.

Чтобы избежать этих проблем при разработке программного обеспечения, как заказчики, так и исполнители, будь то крупная компания или небольшая команда программистов, должны придерживаться методологии обеспечения качества программного обеспечения, состоящей всего из нескольких пунктов.

1. Анализ требований

Еще на этапе формирования технического задания на разработку программного обеспечения необходимо согласовать ключевые вопросы, связанные с механикой работы и составом ключевых компонентов программы. Также стороны должны прийти к взаимному пониманию функциональности создаваемого программного про-

дукта, что позволит добиться принятия работы сразу после демонстрации рабочего образца программы.

2. Анализ кода и сквозное управление

Контроль работоспособности программного кода, наличия ошибок и правильности их обработки должен осуществляться постоянно, в течение всего процесса разработки программного обеспечения. Абсолютно невозможно переложить необходимость поиска проблемных областей кода на плечи тестировщиков, которые занимаются проверкой выполнения основных функций программного обеспечения после основных этапов разработки.

3. Тестирование сеанса

Метод сессионного тестирования, предложенный одним из ведущих специалистов в области программирования Джеймсом Бахом, позволяет провести качественный тест созданного решения. В отличие от технологии поиска «точечных» дефектов кода, во время сессионного тестирования тестер получает свободу действий, пытаясь выявить необычные дефекты, фактически моделируя поведение предполагаемого пользователя.

Наиболее эффективным способом решения проблем в разработке программного обеспечения является обращение к профессиональным «аутсорсерам», предоставляющим услуги IT-аутсорсинга в сегменте разработки программного обеспечения. Ключевыми моментами являются правильный круг ведения, отражающий требования и потребности обеих сторон соглашения, а также установление наиболее оптимальных сроков выполнения заказа. При этом разработчики обязаны доказать обоснованность продления срока реализации проекта, при необходимости, для достижения высокого качества конечного продукта. Только в этом случае качество будет преобладать над временем.

Один из наиболее распространенных способов интенсивного обучения — игровой дизайн (далее — ИП), широко используемый при изучении различных учебных дисциплин.

Их несколько.

1. ИС развивает навыки совместной деятельности, обучает сотрудничеству, т.е. развивается метакомпитенция.
2. Групповая работа объединяет студентов, развивая чувство не только индивидуальной, но и коллективной ответственности.
3. Работа над проектом позволяет студентам развивать аналитический, прогностический, исследовательский и творческий потенциал.
4. В ходе защиты проектов развиваются презентационные навыки, коммуникативная и интерактивная компетентность студентов.

Таким образом, стажеры получают возможность получить действительно практический опыт в решении конкретных задач и попытаться довести решение до реализации в лаборатории.

В этом суть процесса ИС и Его отличие от любого другого процесса принятия решений, основанного на мобилизации коллективного опыта. Игровой дизайн может включать в себя различные типы проектов: исследовательские, поисковые, творческие, прогнозные, аналитические.

Специфика ИП заключается в том, что он представляет собой интерактивный метод, т.е. все проекты разрабатываются в рамках группового игрового взаимодействия, а результаты проектирования (т.е. сам проект, визуально оформленный на листе чертежной бумаги) защищены межгрупповым обсуждением, результаты которого можно определить, во-первых, наиболее разработанный и обоснованный, во-вторых, наиболее представленный проект и как-то поощрить его авторов.

Как известно, важность технологии определяется прежде всего положительными эффектами, которые считаются результатом обучения. Есть несколько из них в игровом дизайне:

- ИС развивает навыки командной работы, учит взаимодействию в команде, т.е. развивается метакомпитенция;
- в процессе защиты проекта развиваются навыки презентации, коммуникативная и интерактивная компетентность обучаемых.

Таким образом, участники взаимодействия в учебном процессе получают возможность получить действительно практический опыт в решении конкретных задач и попытаться довести решение до реализации в лаборатории.

В больших играх, архитектура довольно сложная. Сложные сущности и сложные взаимодействия между классами. Если вы попытаетесь разработать игры с использованием стандартного подхода ООП, вам гарантирована постоянная доработка большого количества кода и сильное увеличение продолжительности разработки.[1,119]

Проблема кроется в наследстве (проблема хрупких базовых классов — ситуация, когда невозможно изменить реализацию типа-предка без нарушения правильного функционирования потомок типов). В поисках решения этой проблемы был разработан компонентный подход (коп). Короче говоря, суть коп заключается в сле-

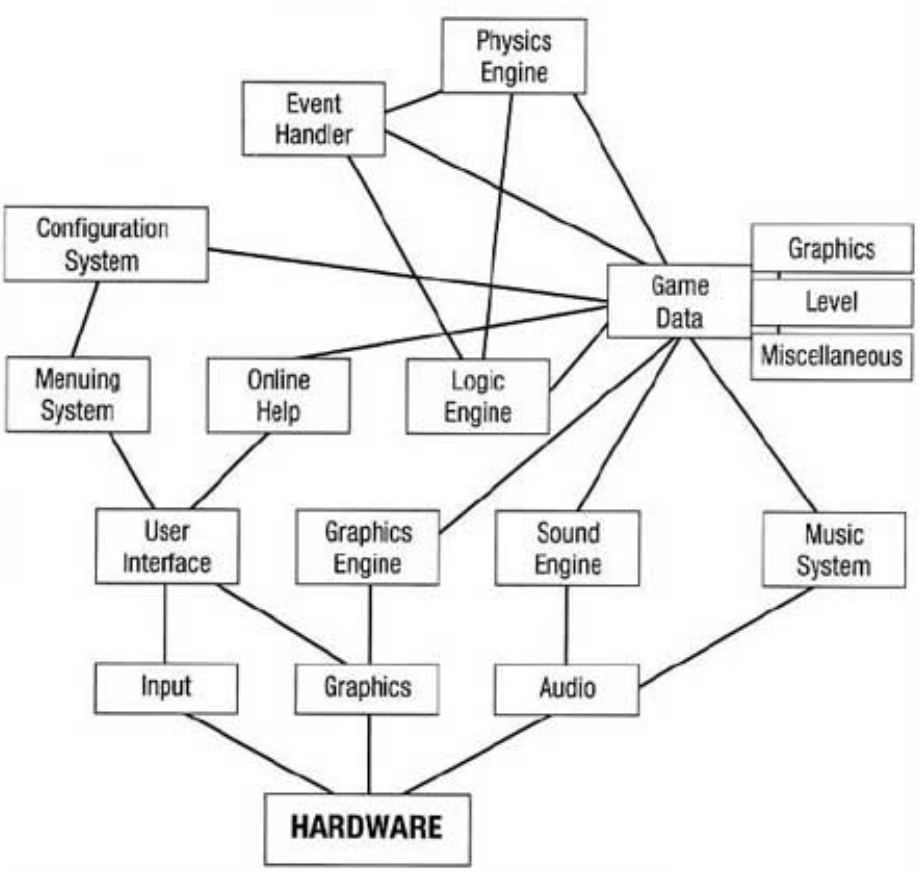

Рис 1. Архитектура Игровая архитектура дизайна

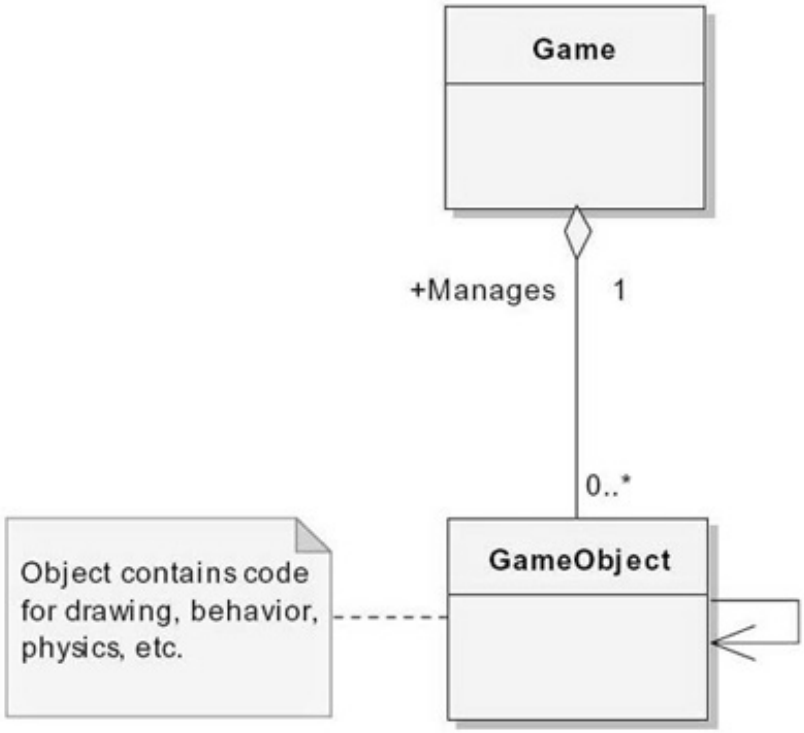

Рис 2. Игра в ООП

дующем: «существует класс контейнера, а также класс компонента, который может быть добавлен в класс контейнера. Объект состоит из контейнера и компонентов в контейнере.» [5, с. 49]

Компоненты немного похожи на интерфейсы. Но интерфейсы позволяют классам иметь общую подпись функций и свойств, а компоненты позволяют создавать общую реализацию классов отдельно. В подходе ООП объект определяется его классом. В подходе CPC объект определяется компонентами, из которых он состоит. Неважно, какой объект. Важно, что у него есть и что он способен сделать.

Пример проблем, с которыми столкнулся id, конечно, не уникален.

Часть проблемы заключается в том, что почти каждая игровая модель основана на принципе объектной ориентации и, следовательно, каждая игра имеет большое количество объектов, которые нужно правильно отобразить и охарактеризовать

Игровые объекты имеют собственное поведение, рисуют себя на экране, а иногда даже говорят сами за себя. Такой подход представляется логичным, и его распространение, по-видимому, связано с универсальным признанием объектно-ориентированной парадигмы. [4, с. 156]

Таким образом, создавая гибкое архитектурное решение, разработчик сокращает время, трудозатраты снижаются.

ЛИТЕРАТУРА

1. Леонтьев В. П. Мир компьютерных игр.— М.: ОЛМА, 2009.— 256 с.
2. Первин С. П. Дети, компьютеры и коммуникации. // Информатика и образование.— 2004. № 4.— С. 17–20.
3. Шапкин А. С. Компьютерная игра: новая область психологических исследований. // Психологический журнал.— 2008. № 1.— С. 79–82.
4. Рейнбоу В., "Компьютерные игры". Энциклопедия.— С.: "Питер" 2005.—732с.
5. Роллингз, Эндрю. "Проектирование и архитектура игр": пер. с англ./ Э. Роллингз, Д. Моррис.— М.: Вильямс, 2006.— 1040с.

Журнал «Современная наука: актуальные проблемы теории и практики»

МИРЭА