УДК 004.451.2

ОБЗОР ПРАКТИК УПРАВЛЕНИЯ ПРОЕКТАМИ В ИГРОВОЙ РАЗРАБОТКЕ

А. В. Шубин[1] [0000-0002-6203-3268], Г. Ф. Сахибгареева[2] [0000-0003-4673-3253],
В. В. Кугуракова[3] [0000-0002-1552-4910]

[1,2,3]*Институт информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета;*

[1]shubin.aleksey.kpfu@gmail.com, [2]gulnara.sahibgareeva42@gmail.com, [3]vlada.kugurakova@gmail.com

Аннотация

Опыт игровых студий показывает, что классические методологии разработки программного обеспечения (ПО) плохо имплементируются в видеоигровую разработку из-за ее интерактивной составляющей, связанной с правильным выстраиванием обратной связи между игрой и пользователем. Кроме того, разработка видеоигр объединяет большое число разработчиков различных сфер, деятельность которых обязана быть согласована в проекте. Несмотря на эти отличия, видеоигры, как и любое другое разрабатываемое ПО, нуждаются в процессе организации команды разработки.

В статье приведён обзор традиционных методологий разработки программного обеспечения, а также модификаций, специализирующихся конкретно на разработке видеоигр. Проведено сравнение наиболее популярных методологий и определено качество их имплементации в студиях разработки видеоигр.

Ключевые слова*: видеоигра, программная инженерия, игровой дизайн.*

ВВЕДЕНИЕ

Развлечения – одна из важнейших частей жизни любого человека: каждый находит свой способ разнообразить свой досуг, отдохнуть, получить новый опыт. Вместе с технологическим прогрессом у человечества появляются новые виды развлечений, такие как книги, кино, аттракционы, видеоигры. На данный момент насчитывается более трёх миллиардов людей, когда-либо приобретших видеоигру, что может говорить об огромной популярности данного вида развлечений,

особенно учитывая, насколько мало времени прошло с момента распространения видеоигр в массы [1].

Видеоигра – это программа, обеспечивающая взаимодействие игроков с игрой и между собой по определенным правилам, а также способное выводить информацию о состоянии игры и реагировать в ответ на действия игроков, которые осуществляются с помощью контроллера.

Видеоигры с самого своего появления не прекращают развиваться, причём происходит это не только в технической части (технологии, компьютерная графика, внешний вид и количество объектов в игре), но и в аспекте игрового процесса (уровень погружения, интерактивность, достоверность). Современные игры давно отошли от концепции «одна игра – одно действие» (Tetris, Pong, Space Invaders), когда весь игровой процесс можно было описать двумя-тремя правилами. Вместо этого игры стали более комплексными, совмещают в себе множество механик или различные типы игрового процесса, более грамотно и осознанно вовлекая игрока.

Созданием игрового процесса в современных играх занимаются такие специалисты, как геймдизайнеры. Они создают правила игры и продумывают логику создания внутриигрового контента. Зачастую в больших компаниях работают целые команды игровых дизайнеров, которые отличаются различной спецификой работы: повествовательная часть игры, внутриигровые локации, правила и системы видеоигры и другие.

Наиболее значимый вопрос при планировании разработки игр – это составление представления о том, какой объем составляет будущий проект. Для того чтобы ответить на этот вопрос, необходимо обратиться к специалистам в сфере разработки игр.

Джесси Шелл, автор «библии» игровой разработки, выделил четыре основных аспекта видеоигры [2]:

Технология – игровой движок, технические характеристики ПО.

История – сюжет, персонажи, цель.

Механика – игровые правила, геймплей.

Эстетика – внешний вид, общее настроение.

Каждый из данных аспектов является ключевым при создании хорошей видеоигры, поэтому при ее разработке геймдизайнер обязан уделить достаточно

внимания каждому из них. Поэтому стоит рассмотреть эти компоненты более подробно.

Технологии – «кости» любой видеоигры. Чаще всего выбор движка будет зависеть непосредственного от типа игрового процесса и характеристик игровых устройств целевой аудитории. Несмотря на это, можно найти и универсальные решения для создания игровых миров, совместимых с большинством игровых устройств, к тому же такие решения будут поддерживать более простой перенос между различными игровыми платформами.

История – это связь между всеми вышеперечисленными аспектами видеоигры. История помогает развивать персонаж игры, даёт ему конкретную, обоснованную цель действий, позволяет игроку понять тему игры. В свою очередь хорошая тема игры вызывает у игрока более сильные чувства, позволяет проникнуться игрой и способствует погружению.

Игровые механики – это «мясо», которое наращивается в движке. Основой любой игры являются правила и возможности игрока в виртуальном мире, и именно это, в свою очередь, определяется реализованными механиками.

Эстетика – это «кожа», внешний вид игры. Благодаря однородной, грамотно выстроенной эстетике игроку можно передать нужное настроение, атмосферу игры и повествования.

Каждый из аспектов, перечисленных выше, хоть и важен для любой видеоигры, но, как показывает опыт, не важно глубоко прорабатывать одновременно все аспекты разом. К примеру, всем известный Tetris – довольно простая по своей сути игра, она не обладает глубокой историей, перипетиями, сюжетными поворотами, в ранних версиях не имела продвинутой графики. При этом игра понравилась многим за счет вовлекающего и репетативного геймплея, который ставит перед игроком непрекращающийся вызов.

Вовлечение – это состояние игрока, при котором он остаётся заинтересованным продолжать процесс игры на долгое время. Данное понятие больше относится непосредственно к игровому процессу, так как требует активных действий со стороны игрока. Из наиболее известных примеров вовлекающих игр можно назвать игры жанра «три в ряд», медитативные игры вроде Journey, а из актуальных на данный момент – Vampire Survivor. И всё это благодаря репетативному, несложному

игровому процессу в связке с получением награды в виде нового контента или части истории.

Погружение – это, в свою очередь, состояние человека, когда он обособлен от физического мира и полностью погружается в выдуманные мир или историю. Поэтому погружение скорее относится к сюжету, передаваемому игрой. Данное состояние можно зачастую наблюдать у людей, читающих захватывающую книгу или смотрящих сериал. В таком состоянии человек попросту не может оторваться от процесса раскрытия сюжета и, сделав перерыв, продолжает думать и планировать что-то в рамках этой книги или сериала.

Названные два понятия имеют различную природу и позволяют игроку задержаться в игре на более долгий промежуток времени, а также способствуют тому, что игрок возвращается и продолжает получать удовольствие от процесса игры.

АНАЛИЗ СУЩЕСТВУЮЩИХ ИГРОВЫХ ПРОДУКТОВ

Для разработки сценария существуют инструменты, призванные облегчить структурирование сюжета. Но средств, способных автоматизировать процесс создания структуры сценария, еще нет. Раннее прототипирование возможно только после создания структуры вручную в программах, которые представляют сюжет в виде связных карточек. Прототипирование предполагает быстрый результат без лишних затрат.

Так как современная наука создания видеоигр до сих пор не может похвастаться чёткими правилами создания идеальной игры, то лучшим вариантом на данный момент является использование лучших практик – правил, открытых опытным путём предыдущими геймдизайнерами. Эти правила сегодня являются неполными, вероятно, даже ненаучными, хотя и помогают молодым разработчикам видеоигр добиться успеха. Такой подход сравнивают с примитивными методами алхимиков до появления Менделеева и, следом, современной химии [2].

Ниже для подбора хороших игровых решений проведен анализ нескольких популярных игр в жанре «приключения». По результатам выделены ключевые аспекты игры, которые делают её более привлекательной для игроков.

Видеоигры категории AAA (например, God of War, Detroit: Become Human, Dying Light 2 и другие) созданы в крупных именитых студиях по разработке игр и

имеют солидный бюджет. Такие игры могут похвастаться крупной командой разработки, проработанным сюжетом, качественными графикой и анимацией, а также выдержанным геймплеем.

Для всех высокобюджетных игр характерна одна общая деталь – современная реалистичная графика (см., например, рис. 1). Зачастую эти игры отличаются соотношением геймплей–сюжет, то есть тем, на какой аспект был сделан больший упор: отзывчивость на действия игрока и свободу действий или на кинематографичность, глубину сюжета и подачу истории. И если одни игры могут похвастаться интересной и разветвлённой историей, но при этом оставляют игроку лишь выбор реплик, то другие имеют тривиальный сюжет, но предоставляют игроку огромную свободу действий, а также содержат вызов игроку, принятие которого требует понимания игры.

Ни одна игра не может похвастаться глубокой проработкой всех ее аспектов, так как в таком случае объём работы будет возрастать в геометрической прогрессии, и велик шанс несогласованности отдельных элементов игры либо вовсе растяжения ее разработки по времени на долгие годы. Именно поэтому зачастую игровые студии стараются найти баланс в этом вопросе и отказываются от проработки отдельных компонентов игры [3].

Рис. 1. Изображение игрового процесса видеоигры God of War (2018)

Взглянув на список известных игр, кроме высокобюджетных, можно заметить, что часто на первых строчках списков популярных игр стоят игры жанра «песочница» (Valheim, Raft, Rust, Project Zomboid и другие). Зачастую разработка таких видеоигр начиналась в маленьких командах либо осуществлялась разработчиками-одиночками, но после обретения популярности команда разработки вырастала до крупной студии.

Инди-игры зачастую строятся на одной геймплейной механике, которая развивается по мере прохождения игры, поэтому остальные аспекты, такие как графика, музыка, сюжет, менее развиты (см., например, рис. 2) либо отброшены вовсе в пользу отполированного игрового процесса.

Рис. 2. Изображение игрового процесса игры Project Zomboid

Если сильнее углубляться в жанр «песочница», то можно заметить, что большая часть игр данного жанра, как и говорилось выше, не имеет глубокого сюжета (Valheim, Raft) или не имеет его вообще (Project Zomboid, Rust), в отличие от упомянутых AAA-игр, но это не мешает им быть популярными среди игроков.

Взглянув на визуальную составляющую, можно заметить, что названные игры не могут похвастаться реалистичной и современной графикой (Project Zomboid), но имеют довольно большую постоянную аудиторию игроков. Так

можно убедиться, что современная графика не является обязательным фактором при создании хорошей видеоигры. Дело в том, что хотя игры названного типа не могут похвастаться большими бюджетами, но при этом, уделив достаточное внимание лишь проработке игрового процесса, малые команды разработки также способны создать игры, не уступающие по популярности играм больших игровых студий.

Так как картинкой и сюжетом данные игры похвастаться не могут, то остаётся последний вариант – непосредственно игровой процесс, и есть то, что цепляет игрока. Но, что самое важное, – это поддержка игрой создания собственной мотивации у игрока, когда цели себе ставит он сам. Такая опция появляется лишь при большой степени свободы действий у игрока, когда у него появляется возможность принимать решения самостоятельно, а не просто идти по указке игры. В данном случае игрок начинает самостоятельно ставить себе цели для исследования мира, составляет план действий, решает какие-либо вопросы, распределяет ресурсы, в конечном итоге достигает максимального комфорта при нахождении в мире игры и, следовательно, самостоятельно создает себе приключение под свой вкус.

Такого игрового процесса сложно добиться простыми планированием и аналитикой. Именно поэтому разработка хорошей видеоигры всегда подразумевает большое количество игровых тестов, корректировок и повторной саморефлексии, а разработчики создают собственные инструменты для оптимизации и балансировки игрового процесса [4, 5].

ОБЗОР МЕТОДОВ РАЗРАБОТКИ ВИДЕОИГР

Разработка игр, как и любое другое производство программных продуктов, обязано использовать современные методологии и практики для эффективной организации труда и сокращения рисков. Крупные компании, большие проекты и сжатые сроки – всё это также актуально для современной разработки игр.

Несмотря на то, что использование современных методологий в большей степени актуально для больших команд, разработчики-одиночки и маленькие инди-студии также используют различные методы и инструменты для протоколирования, прототипирования и разработки. При этом использование методов про-

граммной инженерии в маленьких командах позволяет отслеживать и оптимизировать деятельность, что помогает правильно распределять ресурсы, избавиться от переработок, вести команду разработки в одном направлении, что в конечном итоге позволит создать более качественный продукт.

Каскадная модель

Эта модель развития программного обеспечения использовалась в производстве ранних видеоигр, когда вся разработка была распланирована заранее и каждый этап строго следовал за предыдущим.

Как было сказано ранее, такая модель подразумевает, что каждый этап разработки должен быть полностью закончен, после чего можно переходить к следующему шагу. Данная модель подходит в том случае, если ваша игра распланирована до мельчайших деталей, и по мере разработки её содержание не будет изменяться. Но это и является основным минусом, так как зачастую некоторые решения относительно наполнения игры могут быть спорными или вовсе не подходить игре, а глобальные изменения будут подразумевать полный перезапуск проекта, что невозможно в современных реалиях.

Спиральная модель

Эта модель, в отличие от каскадной, является более подходящей при разработке проектов с изменяющимися на ходу или нечёткими требованиями, так как в отличие от предыдущей модели здесь разработка происходит через несколько повторяющихся этапов (итераций), в конечном итоге дополняя дизайн приложения для конечной разработки (см. рис. 3).

Данная модель, как говорилось ранее, способна легко адаптироваться к изменениям, тем самым она уменьшает риски полной перезагрузки проекта. Также одним из плюсов будет более раннее создание рабочего прототипа, что позволяет видеть прогресс наглядно, а также проверять будущие изменения на основе уже созданного прототипа. К тому же данная модель позволяет подключать всю команду разработки на ранних этапах, а не только лишь по завершению концептуальной разработки. Из минусов можно назвать значительные переработки по сравнению с каскадной моделью, так как в конечном итоге те же самые этапы приходится проходить ни один раз по мере разработки проекта.

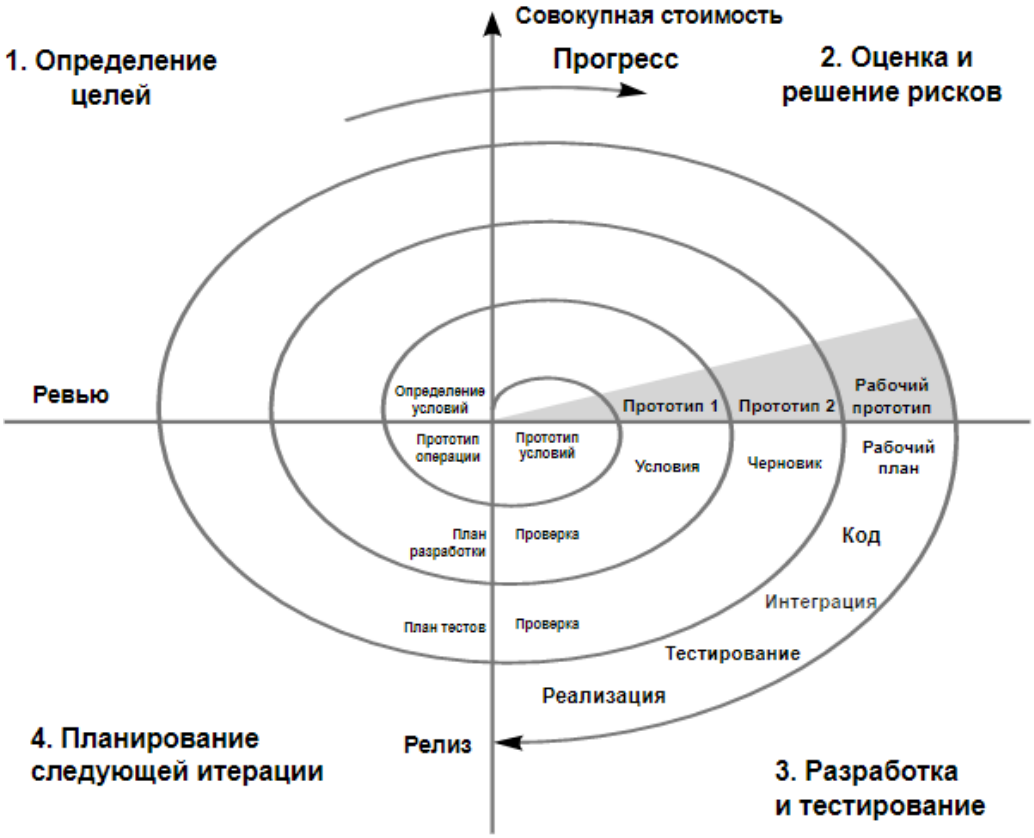

Рис. 3. Схема работы спиральной модели

Итеративная модель

Итеративная модель – другая итерационная модель, но в отличие от спиральной, где большинство требований уже известно и происходит их уточнение, здесь основной упор делается на разработке усовершенствованной версии продукта, после чего подбираются новые требования к следующей итерации.

Данную модель целесообразно использовать в больших проектах, где большие переработки могут значительно замедлить выпуск конечного продукта. Также эта модель не требует полной спецификации требований – они могут уточняться по мере разработки новой версии продукта. Также такая модель позволяет дополнять концепт новыми требованиями, а конфликты между требованиями выявляются раньше.

V-модель

Эта модель является дополнением каскадной модели и отличается от нее наличием параллельного тестирования каждого этапа разработки (см. рис. 4). V-

модель так же, как водопад, является довольно простой и требует полного выполнения предыдущего этапа, но при этом фаза тестирования позволяет скорректировать выполнение каждого этапа. Но, как и в случае с каскадной моделью, остаётся вероятным наличие ошибок на первых этапах разработки. Таким образом, при наличии несоответствия требований их проверка будет происходить слишком поздно, когда сам проект будет уже реализован, что может привести к снижению качества продукта либо, подобно каскадной модели, к перезапуску проекта.

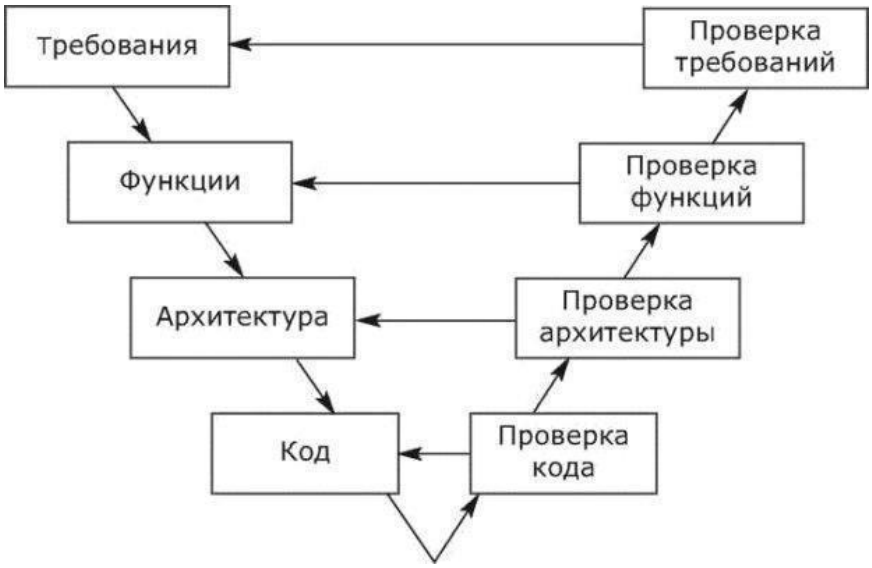

Рис. 4. Схема работы V-модели

Модель большого взрыва

Эту модель сложно назвать методологией как таковой, так как она подразумевает естественную эволюцию проекта, без использования каких-либо практик [6]. Большая часть ресурсов здесь направляется на развитие. Эта методология обычно используется для небольших проектов, когда над разработкой игры работают всего два-три разработчика. При этом такая модель разработки не требует особых знаний и комфортна для малой команды.

Ключевой проблемой данной модели является отсутствие чёткого плана, что может привести к растянутым срокам разработки и долгому поиску решений появляющихся проблем.

Адаптации итерационных моделей

Итерационные модели являются наиболее популярным выбором среди компаний – игровых разработчиков. Используемые ими методологии являются адаптацией классических моделей, модифицированных с целью достижения качественного продукта за счёт более глубокого анализа и увеличения сроков разработки и тестирования видеоигры [7]. Примером можно назвать тестирование потенциальными пользователями или так называемый «ранний доступ», когда игроки могут приобрести доступ к незаконченной версии игры, которая будет исправляться и дополняться в зависимости от мнения этих пользователей. Также применяется такая практика, как этапы Альфа- и Бета-тестирования, когда на первом этапе доступ к ранней версии игры предоставляется закрытому кругу пользователей, заинтересованных в этом продукте и его качестве, а на втором этапе – всем желающим. Следовательно, компании, практикующие данные практики, добавляют такие этапы в свои методологии. Примером служит Blitz Games Studio, где использованы:

Pitch – этап инициации разработки и появления игровой концепции;

Pre-production – создание документации, концепт-артов и развитие игровой концепции;

Main production – непосредственная реализация проекта на игровом движке;

Alpha – тестирование продукта ограниченной группой игроков и исправление наиболее критических ошибок;

Beta – предоставление доступа всем желающим пользователям и итоговая полировка качества продукта;

Master – выпуск игры в полноценную эксплуатацию, конец разработки.

Судя по большому числу вариаций, подобные итерационные модели являются одним из наиболее актуальных выборов при планировании и организации разработки в больших видеоигровых студиях.

Agile-методология

Agile-практика пришла на смену более популярным в своё время методам, таким, например, как водопад или V-модель, и описывает скорее принципы орга-

низации работы команды разработки и практику тесного общения между разработчиками друг с другом и с заказчиком; приоритеты нацелены на удовлетворение клиента, использование методологий с возможностью изменения требований (итерационные подходы), а также саморефлексию [8] – поиск способов быть более эффективными для определения своего поведения в будущем.

На основе принципов Agile были созданы такие методологии, как экстремальное программирование, DSDM, Scrum, FDD и другие [9, 10]. Именно гибкие методологии разработки на данный момент являются наиболее популярными в компаниях при создании любых программных продуктов, так как они позволяют минимизировать риски, вовремя выявлять конфликт требований, оптимизировать работу команды разработки и в перспективе наращивать эффективность их деятельности.

Адаптация Agile под игровую разработку

Одной из интерпретаций Agile-методологии является sdPP (Software Development Project Pattern), адаптированная под создание GDD и разработку видеоигр в целом [11].

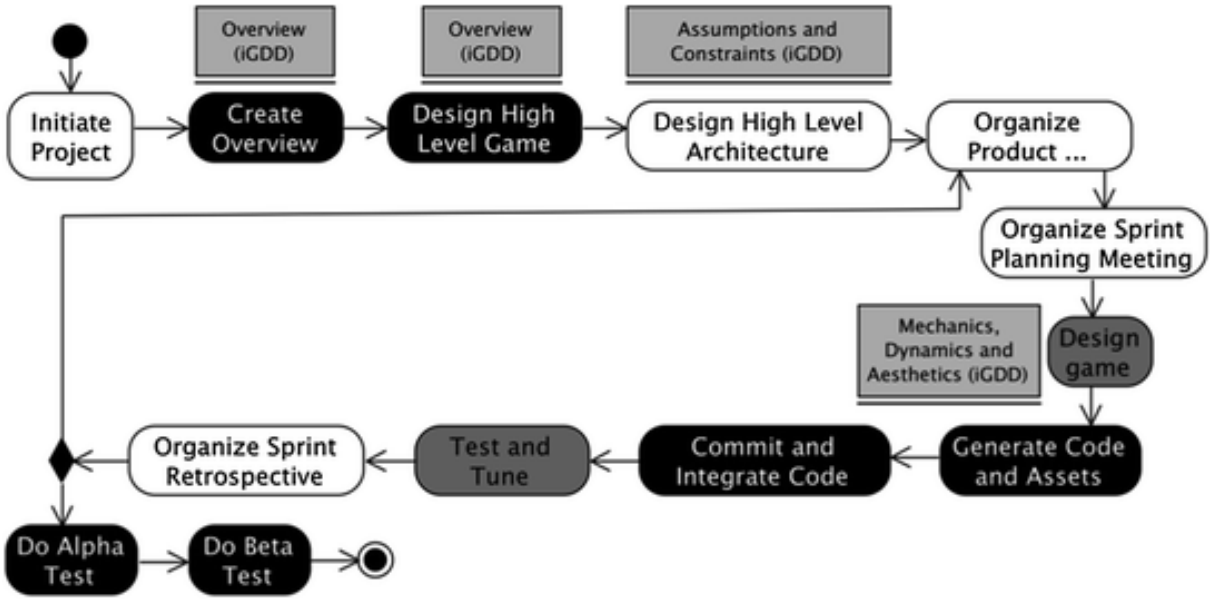

Рис. 5. Схема работы sdPP-модели, адаптированной под разработку видеоигр

На рис. 5 чёрными блоками обозначены новые блоки, относящиеся к разработке видеоигр, а тёмно-серыми – преобразованные блоки стандартной sdPP-модели, также светло-серыми обозначены описательные блоки для некоторых этапов. Ниже представлено описание каждого этапа данной методологии.

CREATE OVERVIEW – этап описания основных аспектов игры: цели, жанра, ответы на различные вопросы (стоит ли она того, какова целевая аудитория, основные механики).

DESIGN HIGH LEVEL GAME – определение главных особенностей игры: game modalities (одиночная игра, многопользовательская игра, игра через интернет, аркадный режим, сюжетный режим), поддерживаемые платформы, тема игры, история и размер планируемого проекта.

DESIGN HIGH LEVEL GAME ARCHITECTURE – описание технической составляющей для определения допущений, ограничений, которые могут появиться в игре. Техническая составляющая содержит: стандарты, соглашения, технологии, ресурсы и архитектуру, выбранную для игры.

DESIGN GAME – разработка главного героя, его действий, эстетики игры, а также включенных игровых механик.

GENERATE CODE AND ASSETS – написание кода и подготовка ассетов (музыка, внутриигровые скрипты, трёхмерные модели, анимации и т. д.)

COMMIT AND INTEGRATE CODE – разработка игры по принципу «получить в любой момент» до версии, которую уже можно было предоставить игрокам.

DO TEST AND TUNE – проверка результатов спринта. Маленькие корректировки могут отполировать игру, но большие изменения остаются на следующий спринт. В конечном итоге команда должна дойти до продукта, который можно будет продать.

DO ALPHA AND BETA TESTS – устранение багов на альфа-тесте, чтобы на бета-тесте игра была проходимой для отслеживания опыта игроков.

Данная модификация была специально спроектирована, чтобы конкретно использовать её при разработке видеоигр с целью уменьшения количества переработок в команде.

В статье [11] описан эксперимент, в котором две сбалансированные команды молодых разработчиков использовали различные подходы при создании видеоигры:

Группа А – использование адаптированной sdPP-модели.

Группа Б – применение стандартной Agile-методологии.

Разработка проектов проходила в течение трёх месяцев, где команды обязаны были реализовать от 10 до 15 функций за 4–5 итераций.

Описанные результаты эксперимента подтверждают, что среднее значение переработок в группе Б превышало более чем в четыре раза среднее количество переработок в группе А.

ЗАКЛЮЧЕНИЕ

Разработка видеоигр – молодая индустрия, основой которой является использование так называемых «лучших практик», когда новые проекты зачастую делают на примере более удачных аналогов. Использование современных методологий разработки каноничного программного обеспечения будет менее эффективным, так как специфика разработки видеоигр делает упор именно на интерактивную составляющую, которую сложно описать и полностью промоделировать на этапе предпродакшена. Сложность в прогнозировании качества принятого решения диктует разработчикам, что необходимо сначала реализовать идею, а затем делать выводы о том, насколько удачно та или иная механика передаёт смысл и синергирует с другими, уже реализованными аспектами игры. Хотя в разработке инди-проектов нередко бывают случаи, когда разработчики внедряют совершенно экспериментальные решения, которые впоследствии становятся удачными среди игроков, такой вариант невозможен для большой команды разработчиков, нуждающейся в стабильном успехе.

Отмеченный аспект обусловлен по большому счёту комплексностью интерактивной составляющей игры: связь игровых механик с повествовательной частью, планомерное развитие механик по мере прохождения, учёт начального уровня подготовки игрока, а также неоднократные тесты жизнеспособности выбранного решения (из-за отсутствия универсального решения). Кроме того, важно уделить должное внимание каждому из четырёх аспектов игры (технология, история, меха-

ника, эстетика), так как недостаточное развитие какого-либо из этих аспектов может привести игрока к диссонансу. Также важен анализ большого количества конкурентных проектов, чтобы на их примере реализовывать собственные идеи.

В заключение можно сказать, что даже каноничные методологии разработки программного обеспечения «в чистом виде» хотя и помогают намного лучше организовать работу, но из-за специфики предметной области требуют дополнительной детализации при разработке. Используя методологии и инструменты, адаптированные под игровую индустрию [12], команда сокращает количество переработок, увеличивает детализацию и качество дизайна проекта, следовательно, качество итогового продукта.

БЛАГОДАРНОСТИ

Работа выполнена за счет средств Программы стратегического академического лидерства Казанского (Приволжского) федерального университета («ПРИОРИТЕТ-2030»).

СПИСОК ЛИТЕРАТУРЫ

1. Global video game consumer population basses 3 billion. URL: https://www.dfcint.com/dossier/global-video-game-consumer-population/ (дата обращения: 20.04.2022).

2. *Шелл Дж*. Геймдизайн. Как создать игру, в которую будут играть все. М.: Альпина Паблишер, 2022. 640 с.

3. Game Project Management and Business Context. URL: https://www.studytonight.com/3d-game-engineering-with-unity/business-context-of-game (дата обращения: 23.05.2022).

4. *Сахибгареева Г.Ф., Кугуракова В.В.* Практики балансирования компьютерных игр // Программные системы: теория и приложения. 2022. Т. 13. №3(54). С. 255–273.

5. *Сахибгареева Г.Ф., Кугуракова В.В., Большаков Э.С.* Генерация и балансирование игровых механик видеоигр // Научный сервис в сети Интернет: труды XXIV Всероссийской научной конференции (19–22 сентября 2022 г., онлайн). М.: ИПМ им. М.В. Келдыша, 2022. С. 455–485.

6. Game Development Lifecycle Models.

URL: https://www.studytonight.com/3d-game-engineering-with-unity/game-development-models (дата обращения: 12.05.2022).

7. *Aleem S., Capretz L.F., Ahmed F.* Game development software engineering process life cycle: a systematic review // Journal of Software Engineering Research and Development. 2016. Vol. 4, No. 6. https://doi.org/10.1186/s40411-016-0032-7.

8. Manifesto for Agile Software Development. URL: http://agilemanifesto.org (дата обращения: 12.05.2022).

9. Ещё раз про семь основных методологий разработки. URL: https://habr.com/ru/company/edison/blog/269789/ (дата обращения: 15.05.2022).

10. Популярные фреймворки для гибкого управления проектами. URL: https://www.wrike.com/ru/project-management-guide/agile-frejmvorki-v-upravlenii-proektami/ (дата обращения: 20.05.2022).

11. *Mitre-Hernández H.A., Lara-Alvarez C., González-Salazar M., Martín D.* Decreasing Rework in Video Games Development from a Software Engineering Perspective // Trends and Applications in Software Engineering. Advances in Intelligent Systems and Computing, 2016. Vol. 405. P. 295–304. https://doi.org/10.1007/978-3-319-26285-7_25.

12. *Шараева Р.А., Кугуракова В.В., Селезнева Н.Э.* Методика упрощения таск-трекинга в проектах игровой индустрии // Программные продукты и системы. 2022. Т. 35. № 3. С. 374–383. https://doi.org/10.15827/0236-235X.139.374-383.

THE CONCEPT OF AUTOMATIC CREATION TOOL FOR COMPUTER GAME SCENARIO PROTOTYPE

A. V. Shubin [0000-0002-6203-3268], G. F. Sahibgareeva[2] [0000-0003-4673-3253],

V. V. Kugurakova[3] [0000-0002-1552-4910]

[1,2,3] *Institute of Information Technologies and Intelligent Systems, Kazan (Volga Region) Federal University, ul. Kremlyovskaya, 35, Kazan, 420008*

[1]shubin.aleksey.kpfu@gmail.com, [2]gulnara.sahibgareeva42@gmail.com,

[3]vlada.kugurakova@gmail.com

Abstract

The experience of game studios shows that classical methodologies of software development are poorly implemented in video game development because of the interactive component of this area, related to the correct creation of feedback between the game and the user. In addition, video game development involves a large number of developers from different areas, whose activities must be coordinated in the project.

Despite these differences, video games, like any other developed software, need a development team organization process. In this article we reviewed traditional software development methodologies, as well as modifications specializing specifically in video game development. The most popular methodologies were compared and the quality of their implementation in video game development studios was determined.

Keywords*: video game, software engineering, game design.*

REFERENCES

1. Global video game consumer population basses 3 billion. URL: https://www.dfcint.com/dossier/global-video-game-consumer-population/, last accessed 2022/04/20.

2. *Schell J. N.* The Art of Game Design: A Book of Lenses, Second Edition. CRC Press. 2014. 600 p.

3. Game Project Management and Business Context. URL: https://www.studytonight.com/3d-game-engineering-with-unity/business-context-of-game, last accessed 2022/05/23.

4. *Sahibgareeva G.F., Kugurakova V.V.* Praktiki balansirovanija komp'juternyh igr // Programmnye sistemy: teorija i prilozhenija. 2022. V. 13. №3(54). P. 255–273.

5. *Sahibgareeva G.F., Kugurakova V.V., Bolshakov E.S.* Generacija i balansirovanie igrovyh mehanik videoigr // Nauchnyj servis v seti Internet: trudy XXIV Vserossijskoj nauchnoj konferencii (19–22 of September 2022, on-line). 2022. P. 455–485.

6. Game Development Lifecycle Models. URL: https://www.studytonight.com/3d-game-engineering-with-unity/game-development-models, last accessed 2022/05/12.

7. *Aleem S., Capretz L.F., Ahmed F.* Game development software engineering process life cycle: a systematic review // Journal of Software Engineering Research and Development. 2016/ Vol. 4, No. 6. https://doi.org/10.1186/s40411-016-0032-7.

8. Manifesto for Agile Software Development. URL: http://agilemanifesto.org, last accessed 2022/05/12.

9. Eshhjo raz pro sem osnovnyh metodologij razrabotki.
URL: https://habr.com/ru/company/edison/blog/269789/, last accessed 2022/05/15.

10. Populjarnye frejmvorki dlja gibkogo upravlenija proektami. URL: https://www.wrike.com/ru/project-management-guide/agile-frejmvorki-v-upravlenii-proektami/, last accessed 2022/05/20.

11. *Mitre-Hernández H.A., Lara-Alvarez C., González-Salazar M., Martín D.* Decreasing Rework in Video Games Development from a Software Engineering Perspective // Trends and Applications in Software Engineering. Advances in Intelligent Systems and Computing. 2016. Vol. 405. P. 295–304.
https://doi.org/10.1007/978-3-319-26285-7_25.

12. *Sharaeva R.A., Kugurakova V.V., Selezneva N.E.* Metodika uproshhenija task-trekinga v proektah igrovoj industrii // Programmnye produkty i sistemy. 2022. V. 35. № 3. P. 374–383. https://doi.org/10.15827/0236-235X.139.374-383.

СВЕДЕНИЯ ОБ АВТОРАХ

ШУБИН Алексей Витальевич – лаборант кафедры программной инженерии Института ИТИС КФУ. Сфера научных интересов – разработка видеоигр, игровой дизайн.

Aleksey Vitalevich SHUBIN – lab assistant at the Department of Software Engineering of the Institute of ITIS KFU. Research interest – videogame development, game design.

e-mail: shubin.aleksey.kpfu@gmail.com

ORCID: 0000-0002-6203-3268

САХИБГАРЕЕВА Гульнара Фаритовна – ассистент кафедры программной инженерии Института ИТИС КФУ. Сфера научных интересов – игровая сценаристика, нарративный дизайн, изучение вопроса эффективности создания сценарного прототипа и возможности автоматизации данного процесса.

Gulnara Faritovna SAHIBGAREEVA – assistant of the Department of Software Engineering of the Institute ITIS KFU. Research interests – game scripting, narrative design, studying the issue of the effectiveness of creating a scenario prototype and the possibility of automating this process.

email: gulnara.sahibgareeva42@gmail.com

ORCID: 0000-0003-4673-3253

КУГУРАКОВА Влада Владимировна – к. т. н., доцент кафедры программной инженерии Института ИТИС КФУ, руководитель НИЛ разработки AR/VR приложений и компьютерных игр. Сфера научных интересов – иммерсивность виртуальных сред, проблемы генерации реалистичной визуализации, различные аспекты проектирования игр, AR/VR, подходы к интерпретации UX.

Vlada Vladimirovna KUGURAKOVA*,* PhD., Docent of the Institute ITIS KFU, Head of Laboratory «AR/VR applications and Gamedev». Research interests include immersiveness of virtual environments, problems of generating realistic visualization, various aspects of game design, AR/VR, approaches to UX interpretation.

email: vlada.kugurakova@gmail.com

ORCID: 0000-0002-1552-4910

СПИСОК УПОМЯНУТЫХ ВИДЕОИГР

1. Tetris [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Тетрис (дата обращения: 20.05.2022).
2. Pong (игра) [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Pong_(игра) (дата обращения: 20.05.2022).
3. Space Invaders [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Space_Invaders (дата обращения: 20.05.2022).
4. Journey on Steam [Электронный ресурс] – Режим доступа: https://store.steampowered.com/app/638230/Journey/ (дата обращения: 20.05.2022).
5. Vampire Survivors on Steam [Электронный ресурс] – Режим доступа: https://store.steampowered.com/app/1794680/Vampire_Survivors/ (дата обращения: 20.05.2022).
6. God of War (игра, 2018) [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/God_of_War_(игра,_2018) (дата обращения: 20.05.2022).
7. Detroit: Become Human on Steam [Электронный ресурс] – Режим доступа: https://store.steampowered.com/app/1222140/Detroit_Become_Human/ (дата обращения: 20.05.2022).
8. Dying Light 2: Stay Human [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Dying_Light_2:_Stay_Human (дата обращения: 20.05.2022).
9. Rust on Steam [Электронный ресурс] – Режим доступа: https://store.steampowered.com/app/252490/Rust/ (дата обращения: 20.05.2022).
10. Raft on Steam [Электронный ресурс] – Режим доступа: https://store.steampowered.com/app/648800/Raft/ (дата обращения: 20.05.2022).
11. Project Zomboid on Steam [Электронный ресурс] – Режим доступа: https://store.steampowered.com/app/108600/Project_Zomboid/ (дата обращения: 20.05.2022).
12. Valheim on Steam [Электронный ресурс] – Режим доступа: https://store.steampowered.com/app/892970/Valheim/ (дата обращения: 20.05.2022).

Материал поступил в редакцию 12 сентября 2022 года