Отчет по использованию парного программирования в проекте "CinemaSync"

Какую задачу реализовывали?

(Юрий Струков и Дмитрий Мищенко) Была реализована логика текстового чата. В парном программировании использовались стили *"ведущий-ведомый"* и "*пинг-понг"*. В роли ведущего находился Юрий Струков из-за более глубоких знаний языка программирования и фреймворка Ruby on Rails в частности.

(Роман Шакун и Николай Тарасевич) Создание и оформление форм для регистрации и входа на сайт. В качестве методики парного программирования была выбрана методика *"пинг-понг"*.

(Роман Шакун и Михаил Нагараев) Оформление дизайна чата. Использовался стиль *"ведущий-ведомый"*, так как один из программистов имеет больший опыт в верстке и мог направлять менее опытного, давать полезные советы.

(Михаил Нагараев и Андрей Новик) Добавление видео. Для выполнения задачи был выбран стиль *"пинг-понг"*.

(Юрий Струков и Николай Тарасевич) Регистрация на сайте. Выбранный стиль программирования – *"ведущий-ведомый",* так как один из программистов имеет больший опыт разработки на Ruby on Rails и мог направлять менее опытного в его начинаниях и давать полезные советы.

Какой стиль показался более приемлемым? Почему?

(Юрий Струков) Как показала практика, стиль *"пинг-понг"* требует примерно равного уровня знаний от обоих программистов. А т.к. в нашем случае это оказалось не совсем так, то и результат получился не самым удовлетворительным. Стиль *"ведущий-ведомый"* мне понравился больше, т.к. здесь, на мой взгляд, большую роль играет сам факт присутствия. Это помогает меньше отвлекаться от решения задач, а также быстрее и проще объяснять некоторые моменты.

(Роман Шакун) Стиль *"ведущий-ведомый"* лично для меня более приемлемый , т.к использование этого стиля помогло углубить знания в верстке, так как более опытный программист давал советы и рекомендации, что помогло справляться с проблемами очень эффективно. Стиль *"пинг-понг"* не очень понравился, так как приходилось ждать пока напарник напишет тесты, из-за чего тратилось время, и само написание тестов оказалось очень скучной задачей.

(Николай Тарасевич) Стиль *"ведущий-ведомый"* мне понравился больше, так как в процессе работы я узнал несколько интересных для себя вещей, которые, уверен, пригодятся мне в будущем. В случае же со стилем *"пинг-понг"* пришлось изучать фреймворки для тестирования в Rails с нуля, что сказалось на времени работы. Но в перспективе данный стиль очень хорош.

(Дмитрий Мищенко) Стиль *"пинг-понг"* оказался достаточно сложным в случае, когда участники обладают разным уровнем знаний языка программирования и инструментов тестирования. Было очень сложно писать тесты, скорость разработки была достаточно невысокой. В случае *"ведущего-ведомого"* данной проблемы уже не возникло, ускорился поиск решений в сети, менее опытный участник (в данном случае я) получил новые знания от более опытного.

(Андрей Новик) Стиль “*пинг-понг*” для меня показался приемлемым. Главной причиной выбора данного оказалась моя любознательность. К тому же это была очень хорошая возможность изучения инструментов тестирования.

(Михаил Нагараев) И стиль *“ведущий-ведомый”,* и стиль *“пинг-понг”* меня устраивают. В первом случае мне понравилось быть наставником, ведь когда помогаешь – сам прогоняешь в памяти уже пройденное и крепче усваиваешь это. Второй случай – писать реализацию по тестам очень удобно, ведь зачастую сразу видно место, в котором что-то идет не так.

Что получилось? Положительные впечатления.

(Юрий Струков) Получилось выполнить некоторые поставленные задачи быстрее, чем если бы я решал их один. Также удалось прояснить не совсем понятные моменты и ликвидировать некоторые пробелы в знаниях как у меня, так и у моих напарников.

(Роман Шакун) Получилось углубить свои знания в отдельных областях и значительно продвинуться в реализации требуемых задач. А также парное программирование позволило развить навыки работы в команде.

(Николай Тарасевич) Благодаря совместной работе было значительно сокращено время на поиски некоторых решений для реализации задач. Плюс в паре гораздо веселее работается. В общем задачи были выполнены без больших временных затрат. Ощутил плюсы TDD.

(Дмитрий Мищенко) Целью парного программирования являлась реализация функционала текстового чата на сайте. Цель была полностью реализована. Нельзя сказать, что скорость разработки “с нуля” значительно возросла, но поиск решений проблем, возникающих в процессе разработки, был заметно более быстрым, быстрее находились новые функциональные решения, исправлялись опечатки.

(Андрей Новик) Поставленная задача была реализована полностью. В самом начале скорость разработки совсем не увеличилась, а даже наоборот упала. Но после практики и получения опыта от товарища, мы начали получать хороший результат. Мне кажется, что в долгосрочной перспективе парное программирование покажет себя очень хорошо. В процессе работы с более опытным программистом было получено огромное количество опыта.

(Михаил Нагараев) Все поставленные задачи были выполнены, причем в достаточно короткие сроки, что порадовало. Понравилось то, что в процессе работы помимо моих решений предлагались и альтернативные, которые зачастую были очень годными и внедрялись.

Что не получилось? Как можно это поправить?

(Юрий Струков) Как уже писалось выше, разный уровень знаний напарников не позволил в полной мере реализовать стиль *“пинг-понг”*. И поэтому для таких случаев в качестве решения могу предложить как можно чаще по возможности использовать стиль *“ведущий - ведомый”*.

(Роман Шакун) Были небольшие неудобства со стилем “пинг-понг” , так как приходилось ждать пока твой напарник напишет тесты и только потом приступать к работе. Также возникали некоторые разногласия о дизайнерских решениях, но все они , в конечном итоге, сводились к консенсусу.

(Николай Тарасевич) Порой возникали разногласия по поводу того, как реализовать ту или иную функциональность, что разочаровывало. Но в процессе обсуждения и приведения положительных сторон своих решений, выбиралось наиболее оптимальное. Не получилось написать идеальные тесты.

(Дмитрий Мищенко) Как и в процессе любой парной работы, возникали ситуации, когда один из участников отвлекался, занимался другими делами. Не всегда выходило так, что один участник знал решение, которого не знал другой, и все сводилось к совместному поиску решений в Интернете.

(Андрей Новик) При парном программировании я постоянно отвлекался и вспоминал клевые истории, которыми просто обязан был поделиться с напарником. Так же поиск в интернете не всегда приводил к нужному результату. Но самым сложным и неприятным в работе оказалось тестирование в Ruby.

(Михаил Нагараев) Иногда отвлекались на сторонние вещи, что плохо сказывалось на работоспособности. Медленно въезжали в сферу тестов на Ruby.

Будете ли использовать парное программирование для других задач?

(Юрий Струков) Скорее да, чем нет. Особенно в случаях решения задач, которые требуют сильной концентрации внимания и быстрого выполнения.

(Роман Шакун) Да, ведь плюсов от парного программирования гораздо больше чем минусов, и минусы не так уж критичны.

(Николай Тарасевич) Скорее да, чем нет. Единственное, что мешает – одна клавиатура на двоих, в большинстве случаев.

(Дмитрий Мищенко) Эффективность парного программирования проявляется только в случае, когда участники пары обладают приблизительно одинаковым уровнем знаний технологий, используемых при разработке проекта. Парное программирование будет использоваться и в дальнейшей разработке данного проекта, так как позволяет ускорить изучение технологий языка программирования менее опытными участниками команды.

(Андрей Новик) Несмотря на достаточно большое количество минусов, у парного программирования оказалось пару весомых плюсов. Я бы использовал парное программирования в тех ситуациях, когда на выходе необходимо будет получить код с наименьшим количеством ошибок. Также парное программирование помогло обменяться опытом с коллегами.

(Михаил Нагараев) Да, несомненно. Код получается гораздо более чистым, в процессе работы могут появиться нестандартные и гораздо более гибкие решения, что, конечно, очень хорошо сказывается на качестве конечного продукта. Используя TDD гораздо проще отследить ошибки, что упрощает разработку.