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 "*удаленное парное программирование"*. 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 *"водитель-штурман"*. Так она показалась нам наиболее приемлемой для нашей задачи. Один из участников отвечал за дизайн форм (подбирал стиль формы, цвета, шрифты) и создавал макет, а второй участник реализовывал все это в коде.

(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 *"удаленное парное программирование"*.

(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) Стили *“ведущий-ведомый”* и *“удаленное парное программирование”* по сути своей довольно схожи, однако 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 за счет наличия под рукой листика и ручки и возможности более точно донести до напарника какие-либо сведения с помощью эмоций и жестов.

(Роман Шакун) Оба стиля (*“водитель-штурман”* и *“ведущий-ведомый”*) оказались вполне приемлемыми и очень даже эффективными. При использовании первого стиля затрачивалось гораздо меньше времени на выполнение задачи и значительно увеличивалась производительность. Использование второго стиля помогло углубить знания в верстке, так как более опытный программист давал советы и рекомендации, что помогло справляться с проблемами очень эффективно.

(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 *“водитель-штурман”* огромного удовольствия от работы не получил, но он тоже показался мне неплохим. Главное, что продуктивность была на высоте.

(Дмитрий Мищенко) Стиль *“удаленное парное программирование”* без использования дополнительных инструментов (редакторов кода с совместным редактированием и т.д.) с самого начала показал свою несостоятельность в случае, когда один участник обладает более широкими знаниями, чем второй. В результате скорость разработки не повысилась, так как код писал один участник. 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 "*удаленное парное программирование*" для меня показался приемлемым. Главное преимущество данного стиля заключается в удаленности. Также, благодаря постоянному наблюдению за написанием кода, он позволяет избегать ошибок на самом раннем этапе.

(Михаил Нагараев) И стиль *"ведущий-ведомый",* и стиль *"удаленное парное программирование"*) меня устраивают. В первом случае мне понравилось быть наставником, ведь когда помогаешь – сам прогоняешь в памяти уже пройденное и крепче усваиваешь это. Второй случай – своя клавиатура под рукой и работа на своем удобном месте.

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.

(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) Благодаря парному программированию удалось существенно уменьшить время, уходящее на поиск решения проблемы. В процессе наблюдения за 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) Во время *“удаленного парного программирования”* имели место быть моменты, когда один или оба участника отвлекались на не связанные с решением задач предметы. Из-за этого снижалась скорость решения этих самых задач и тратилось время на то, чтобы заново сконцентрироваться. В качестве решения могу предложить как можно чаще по возможности заменять удаленное парное программирование стилем *“ведущий - ведомый”*.

(Роман Шакун) Иногда возникали разногласия о дизайнерских решениях, но все они , в конечном итоге, сводились к консенсусу. Нет очевидного решения данной проблемы , так как “на вкус и цвет”.

(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.

(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) Из-за “наличия клавиатуры” только у одного человека, результат совместной работы 2 человек оказался хуже, чем результат раздельной работы. В следующий раз необходимо будет использовать сервисы, позволяющие редактировать код параллельно.

(Михаил Нагараев) Иногда отвлекались на сторонние вещи, что плохо сказывалось на работоспособности. В случае с *“удаленным парным программированием”* было сложнее помочь напарнику показать решение возникавших проблем.

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.