1Лабораторная работа №4

на тему : «Парное программирование»

Виды ПП

Существует несколько **стилей** парного программирования:

- **Ведущий-ведомый**. В этом случае на втором месте, после разработки часто стоит обучение менее опытного программиста, как правило, именно он непосредственно пишет код. Второй программист более опытный, он подсказывает, направляет, дает рекомендации;
- **На равных**. В этом случае работают два примерно одинаковых по опыту разработчиков, время от времени меняющихся местами;
- **Водитель-штурман**. В этом случае программисты выбирают разные роли. Один пишет код, разбирается в деталях, а второй занимается архитектурой кода, решением логических задач, рисует схемы;
- **Пинг-понг**. Один программист пишет тест, а второй – реализацию под него. После происходит смена ролей;
- **Удаленное ПП**. Единственным минусом такого стиля является то, что нельзя ткнуть пальцем в экран. А если серьезно, то в этом случае можно использовать разные инструменты: например, давать партнеру возможность одновременно с вами писать код или же только видеть ваш экран.

Самый большой скепсис вызывает вопрос: Не будет ли разработка идти медленней, когда два программиста занимаются одной задачей?

Исследования показывают, что работа в паре делается либо с такой же скоростью, как и по одиночке, либо немного (15%) медленнее. Зато код получается намного качественней, содержит меньше ошибок (60%) и технических долгов.

Бонусы от парного программирования

1. **Обмен опытом:** Часто бывает, что сидя в паре вы узнаете про пару новых горячих клавиш, интересные утилиты для ускорения работы. В любом случае, наблюдая за тем, как программируют другие вы сами постоянно учитесь.

2. **Знания о системе**: Постоянная смена пар способствует распространению знаний о разных частях системы внутри команды. Это дает возможность понимать как система развивается, улучшать дизайн системы, не дублировать логику.
3. **Коллективное владение кодом**: Когда все участвуют в написании всех частей системы, то не может идти речи о персональном владении классом или сборкой.
4. **Наставничество**: Все мы когда-то начинали программировать. Как показала практика самое простое вливание в проект происходит в процессе парного программирования.
5. **Больше общения**: Общение внутри команды помогает выстраивать доверительные отношения. Стендапы и ретроспективы добавляют в общения в повседневную работу, но это не сравнить с возможностями парного программирования.
6. **Стандарты кодирования**: Сидя в паре, постоянно передавая клавиатуру и меняя пары, программисты распространяют знания о том, какие стандарты кодирования приняты на проекте. Вам уже не понадобится прикручивать автоматические инструменты для проверки качества кода.
7. **Улучшение дисциплины**: Сидя в паре, хочется показать свою заинтересованность и уровень подготовки партнеру. И довольно трудно временно переключиться на соц. сети, чтобы полистать последние забавные картинки.
8. **Сопряжение потока**: Один программист спрашивает у другого «Что мы сейчас решаем?» и они оба начинают погружаться в задачу. Такой подход может приводить к сопряжению состояния потока, что увеличивает продуктивность в разы.
9. **Меньше прерываний**: В паре вам приходится меньше прерываться на сторонние факторы, т.к. время двух человек ценнее, чем одного, их работа становится в 2 раза дороже.

Ошибки при парном программировании

1. **Наблюдай за Мастером**: Это происходит, когда в паре есть программист, который считает себя (или даже является) гуру в своей области. Вопросы менее опытного разработчика о коде, который генерируется Мастером, не получают ответа. Возможен вариант, когда его постоянно посылают почитать в Google. Мастер не спешит отдавать клавиатуру напарнику, а когда тот добирается до нее, Мастер теряет всякий интерес к процессу.
2. **Диктатор**: Один из разработчиков в паре всегда занимает жесткую ультимативную позицию по поводу всех решений, которые касаются текущих задач. В такой ситуации не может идти речи о взаимной помощи или обучении в паре.
3. **Сходи за кофе**: Пара садится за компьютер. Один из разработчиков берет клавиатуру и начинает писать код. Говорит напарнику: «Пока я пишу код, ты сходи и

налей нам кофе». Это нарушает базовую идею о взаимной вовлеченности программистов в процесс.

4. **Молчаливые партнеры**: Напарники не общаются друг с другом и не комментируют свои действия и решения по ходу работы. При отсутствии обратной связи смысл пары теряется.
5. **Разделение задач за одним столом**: Программисты садятся в пару, берут два компьютера за одним столом (настольный и ноутбук) и начинают параллельно работать.
6. **Неудобно сидеть**: Самая частая причина усталости при работе в паре — неудобное положение клавиатуры и монитора для того, кто сейчас «водитель». Когда клавиатура переходит от одного программиста к другому, получивший ее не перемещается в центр стола, а нагибается к клавиатуре, тем самым создавая себе трудности при работе.
7. **Партнер занят своим делом**: Один из партнеров во время работы в паре отдаляется от места работы, проверяет свою почту и т.д.
8. **Свои настройки окружения**: Каждый раз, когда управление переходит от одного партнера к другому, начинается перенастройка окружения: закладок, шрифта и т.д.
9. **Свой стиль**: Каждый из партнеров придерживается своих стандартов кодирования, что вызывает бурные дискуссии и ужасно отформатированный код.

В рамках лабораторной рекомендуется попробовать следующие стили:

1. Ведущий-ведомый (или На равных).
2. Пинг-понг.

Написать свои впечатления о парном программировании:

1. Какую задачу реализовывали?
2. Какой стиль показался более приемлемым? Почему?
3. Что получилось? Положительные впечатления.
4. Что не получилось? Как можно это поправить?
5. Будете ли использовать ПП для других задач? Каких?