🤖 Сдвиг узких мест в разработке с ИИ

🚀 Сдвиг узких мест в разработке с ИИ

Ключевые тезисы:

- С появлением ИИ-инструментов, таких как Claude, **узкие места в разработке сместились**: написание кода, тестирование и рефакторинг перестали быть лимитирующими факторами.
- **Командные процессы и нормы, которые раньше работали, теперь могут "тихо перестать работать"** и требуют пересмотра.
- Новыми узкими местами становятся **верификация, ревью кода и поддержка** из-за возросшей скорости разработки.
- **Роли в командах размываются**, и ИИ становится инструментом для заполнения пробелов в навыках.
- **Ключевой принцип:** то, что служило вам раньше, может больше не работать. Необходимо постоянно пересматривать процессы.

🔄 Сдвиг узких мест

Раньше инженерное время было дорогим ресурсом, и все процессы (планирование, ревью) были построены вокруг его защиты. Теперь, когда **кодирование перестало быть узким местом**, изменились и процессы.

Примеры старых узких мест, которые исчезли:

- Написание кода
- Написание тестов (TDD стало "вкусным", а не "брокколи")
- Рефакторинг (больше не нужно жертвовать временем на фичи)

Новые узкие места:

- ✅ **Верификация** — проверка корректности кода из-за возросшего объема изменений.
- ✅ **Кто делает ревью?** — как справляться с потоком изменений.
- ✅ **Поддержка (Maintenance)** — как управлять стоимостью поддержки возросшего объема кода.

🛠️ Переписанные командные нормы

Из-за сдвига пришлось пересмотреть ключевые процессы команды Claude Code:

- **Планирование:** Глубокие дизайн-документы уступили место обсуждениям прямо в PR или прототипах. **Строить стало дешево, спорить — дорого.**
- **Технические дискуссии:** Теперь разрешаются кодом. Вместо долгих споров на доске можно быстро сгенерировать несколько вариантов реализации PR для сравнения.
- **Прототипирование:** Раньше был риск, что "сырой" прототип пойдет в продакшн. Теперь с Claude прототип можно быстро доработать до production-качества.
- **Ревью кода:** Используется **Claude Code Review**. ИИ отлично справляется с проверкой стиля, поиском очевидных багов и проверкой на соответствие спецификации. **Человек остается в цикле** для:
  - Юридических проверок и оценки рисков.
  - Чувства продукта (product sense) и вкуса.
- **Онбординг:** Код становится главным источником истины. Новые члены команды (и менеджеры) могут погружаться в кодбазу с помощью Claude, задавая вопросы о контексте и архитектуре.
- **Состав команды:** Акцент смещается на два типа профилей:
  1. **Креативные строители с чутьем продукта (product sense).**
  2. **Эксперты с глубокими системными знаниями.**

🎯 Удвоение внимания к верификации и продукту

Поскольку скорость выросла, **качество и уверенность в изменениях стали критически важны.**

Стратегия "Shift Left":

> *Лучше, чем найти баг у пользователя, — найти его самому. Лучше, чем найти его самому, — чтобы его автоматически нашла система.*

Как развивать чутье продукта (Product Sense):

1. **Dogfooding (Antfooding в Anthropic):** Постоянно используйте продукт, который вы создаете. Это особенно важно для менеджеров, которые могут оторваться от кода.
2. **Итерации и выпуск:** Быстро выпускайте, получайте обратную связь.
3. **Общение с клиентами:** Выходите за пределы метрик и дашбордов, чтобы понять реальные проблемы пользователей.

👥 Роли размываются, менеджеры возвращаются к коду

Заполнение кросс-функциональных пробелов:

- Дизайнеры используют Claude для внесения мелких правок в код, ускоряя цикл обратной связи.
- Инженеры используют Claude как партнера по контент-дизайну.

Новая роль менеджера:
На команде Claude Code **каждый менеджер сначала был индивидуальным contributor (IC)**. Это позволяет:

- Лучше понять работу команды изнутри.
- Вернуть себе "часы творца" (maker hours) и не терять связь с кодом.
- Онбордиться в кодбазу с меньшими затратами, не отвлекая инженеров.

🏗️ Внедрение изменений: баланс сверху вниз и снизу вверх

Общие принципы (сверху вниз):

1. Каждый член команды использует Claude Code и Co-work.
2. **Claude-ify everything:** Если это может сделать Claude, пусть Claude это делает.
3. Явное разрешение убивать устаревшие процессы.

Свобода для команд (снизу вверх):
Команды сами решают, как именно внедрять Claude в свои рабочие ритуалы (планерки, триажи, воркфлоу).

📊 Сигналы успеха

Какие метрики показывают, что изменения работают:

- ⬇️ **Время онбординга** сокращается.
- ⬇️ **Время цикла PR** (но важно разбивать его на этапы, чтобы найти новые узкие места, например, в CI/CD).
- ⬆️ **Количество коммитов с помощью Claude** (на команде почти 100% коммитов создаются с помощью ИИ).
- **Важно:** Измерять не только скорость, но и реальное улучшение продукта или качества.

❓ Открытые вопросы и упражнение

Вопросы, над которыми еще работают:

- Нужны ли отдельные команды под iOS и Android, если инженеры могут работать на всех платформах с помощью Claude?
- Как далеко можно зайти в автоматизации ревью?
- Как обеспечить продуктивность всех ролей в условиях их размытия?

Упражнение для вашей команды:

> *Выберите самый "шумный" или затратный рабочий процесс или встречу в вашей команде. Спросите: "Все еще служит ли это своей цели?" и "Может ли Claude помочь нам с этим?". Действуйте шаг за шагом.*

Выводы

Эра ИИ требует **гибкого мышления роста (growth mindset)**. Команды должны постоянно **аудитировать и пересматривать свои процессы**, так как узкие места смещаются, а возможности автоматизации растут. Ключ — не просто увеличить throughput, а сохранить качество, уверенность в изменениях и сфокусироваться на создании ценности для пользователя.
