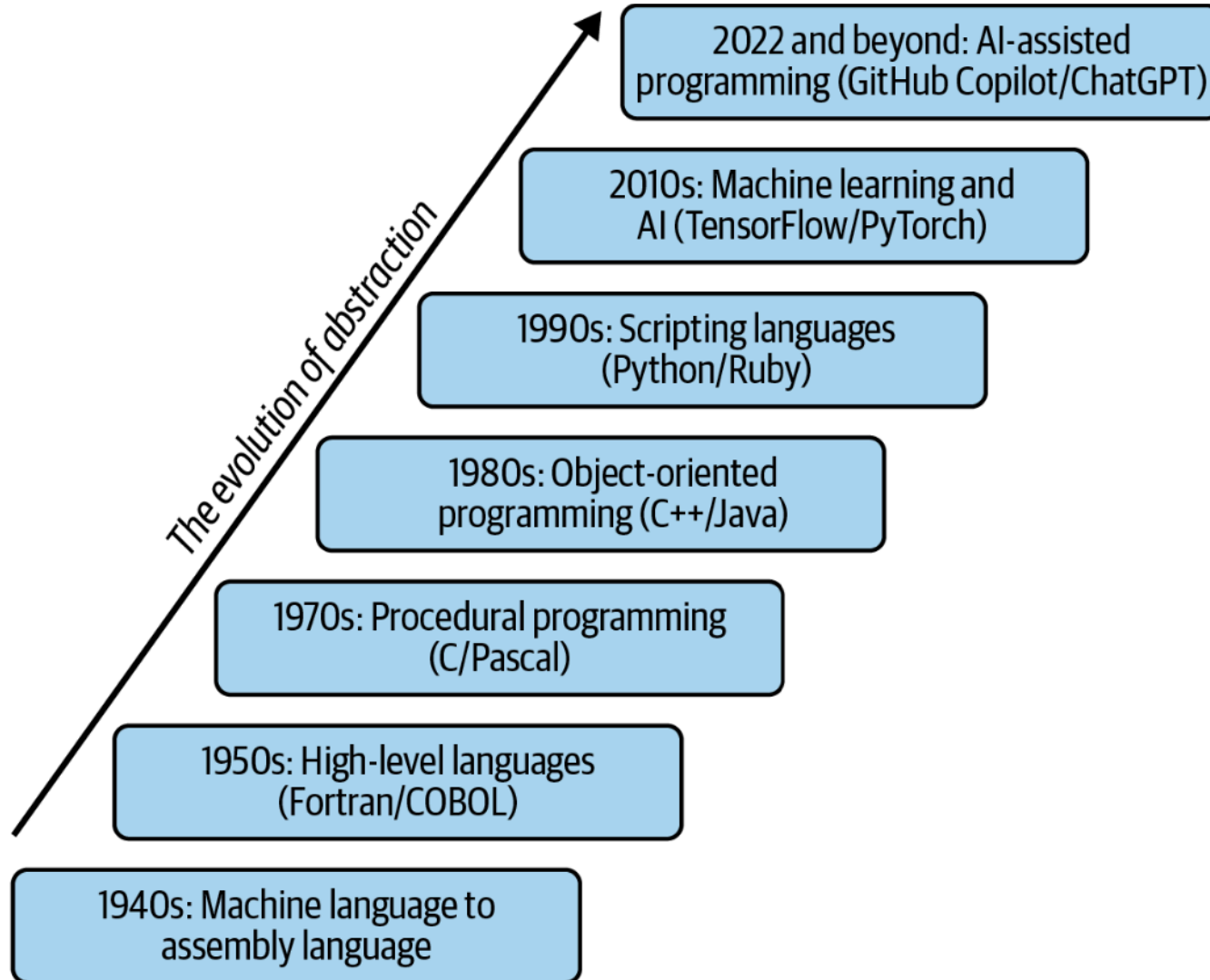




**Практика применения AI-ассистента**

# Эволюция разработки



Рутинные детали скрываются и разработчики могут сосредоточиться на самом важном.

Получают новые возможности, увеличивают технологический рычаг.

# Разработчики говорят о приросте производительности



+

88%

Исследования GitHub Copilot\*



91%

Опрос в Сбере. Июль-Август 2024

\*<https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/>

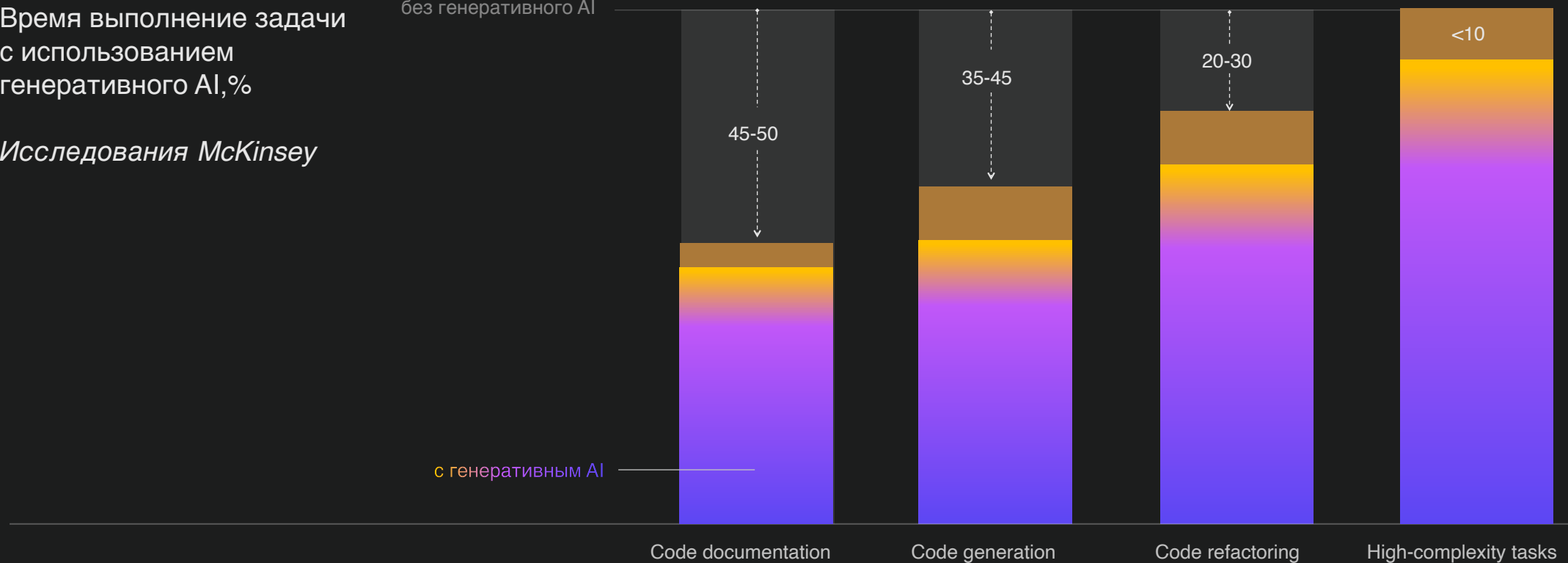
# Помощь AI неоднородна и зависит от типа задач

Время выполнение задачи  
с использованием  
генеративного AI, %

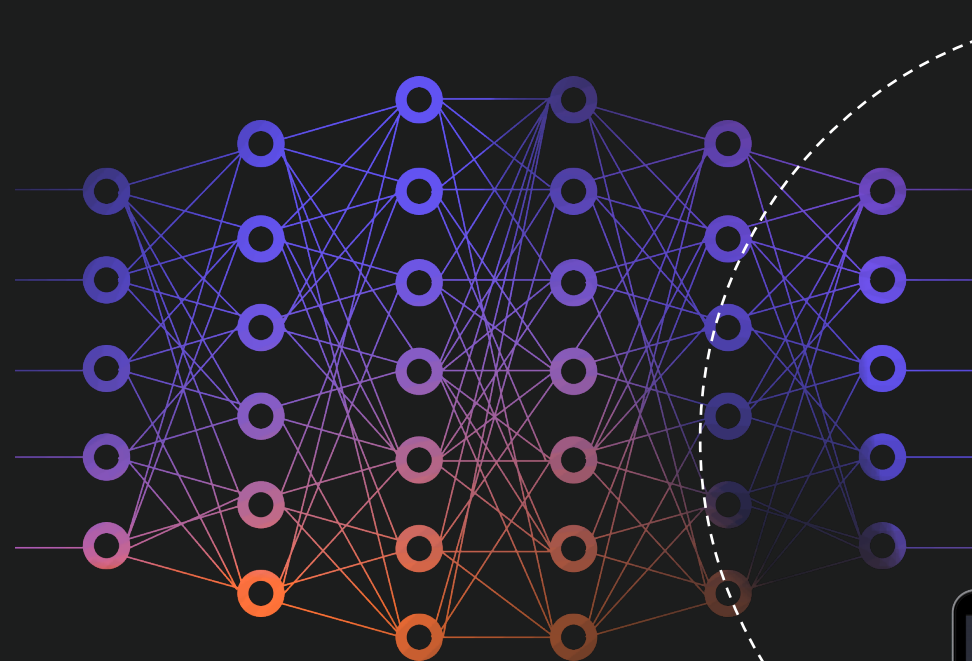
*Исследования McKinsey*

без генеративного AI

с генеративным AI



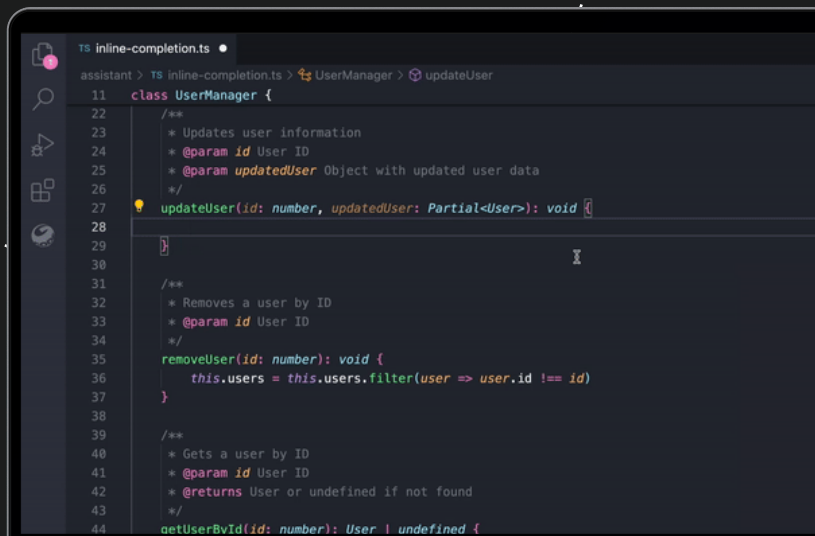
# Что такое AI-ассистент в программировании



+

 **GitHub**

```
require( TEMPLATEPATH/js/scripts/jquery.js )
$render = $document.getElementById( 'render' );
$module = array( 'module' => 'module', 'options' => 'options' );
$stopmenu = false; $subnav = false; $sidenav = false;
Main Menu
if ( $default_menu_style == 1 or $default_menu_style == 2 ) :
    $module->params = "menutype=$menu_name&showallchildren=$showallchildren";
    $stopmenu = $render->render( $module, $options );
    $menuclass = 'horiznav';
    $stopmenuclass = 'top_menu';
elseif ( $default_menu_style == 3 or $default_menu_style == 4 ) :
    $module->params = "menutype=$menu_name&showallchildren=$showallchildren";
    $stopmenu = $render->render( $module, $options );
    $menuclass = 'horiznav';
    $stopmenuclass = 'top_menu';
SPLIT MENU NO SUBS
elseif ( $default_menu_style == 5 ) :
    $module->params = "menutype=$menu_name&startlevel=$startlevel&endlevel=$endlevel";
    $stopmenu = $render->render( $module, $options );
    $menuclass = 'horiznav';
```



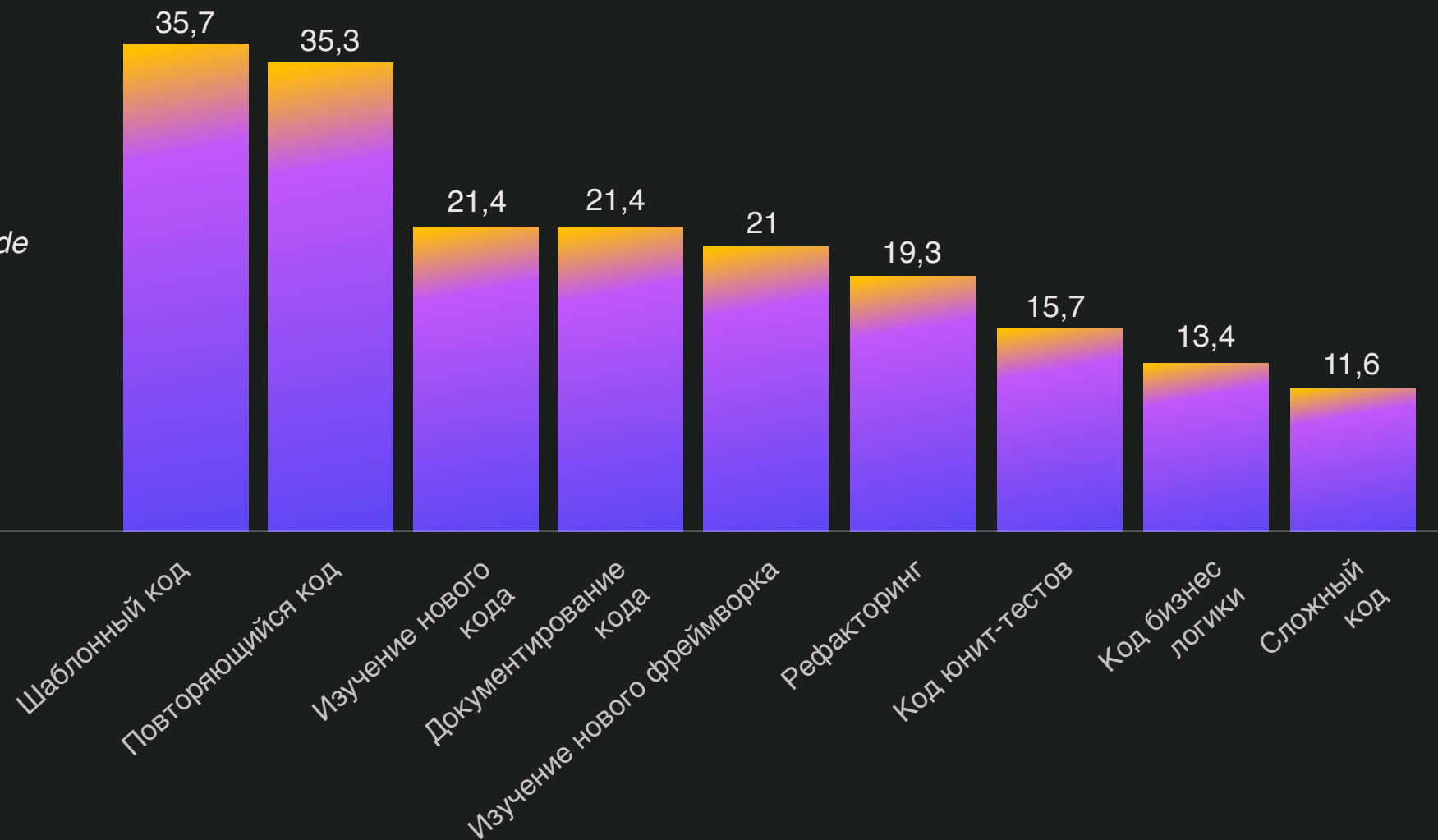
```
TS inline-completion.ts •
assistant > TS inline-completion.ts > UserManager > updateUser
11 class UserManager {
22
23     /**
24     * Updates user information
25     * @param id User ID
26     * @param updatedUser Object with updated user data
27     */
28     updateUser(id: number, updatedUser: Partial<User>): void {
29
30
31
32     /**
33     * Removes a user by ID
34     * @param id User ID
35     */
36     removeUser(id: number): void {
37         this.users = this.users.filter(user => user.id !== id)
38     }
39
40     /**
41     * Gets a user by ID
42     * @param id User ID
43     * @returns User or undefined if not found
44     */
45     getUserById(id: number): User | undefined {
```

MacBook Pro

# Это подтверждается и нашими результатами

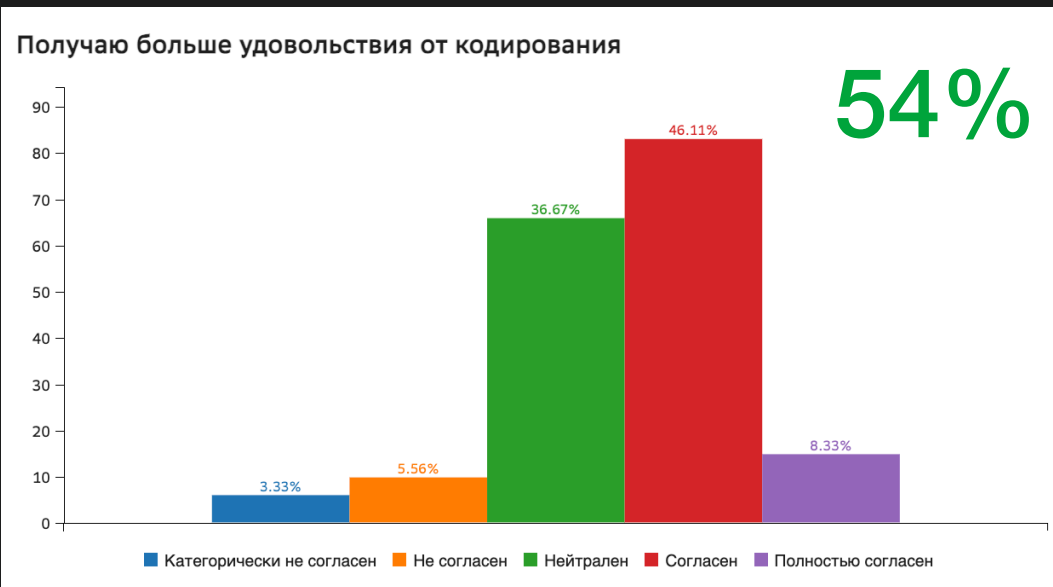
Экономия времени  
в различных случаях  
с Gigacode по оценке  
пользователей, %

Исследования Gigacode

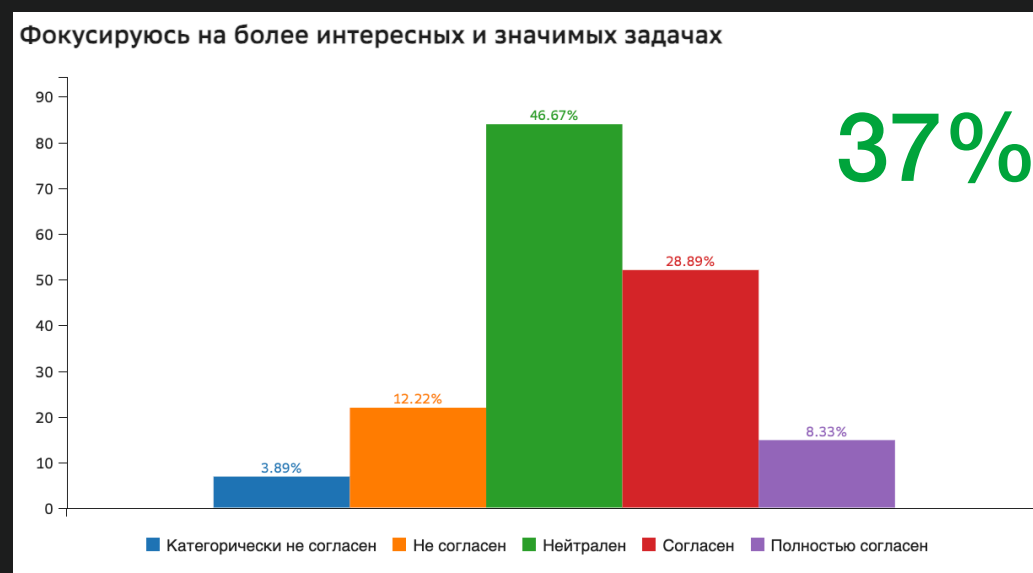
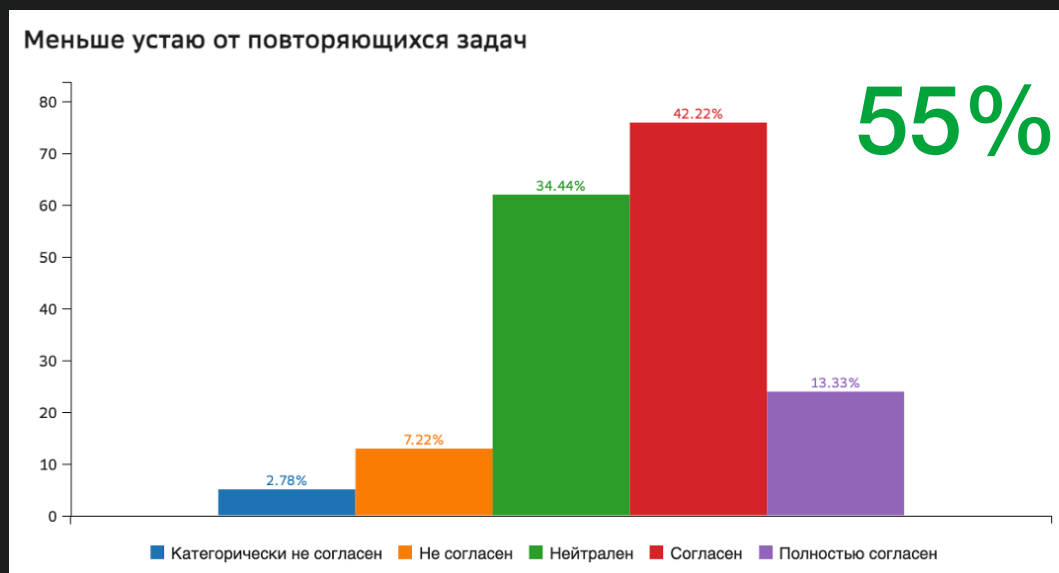


# GigaCode повышает удовлетворенность

7



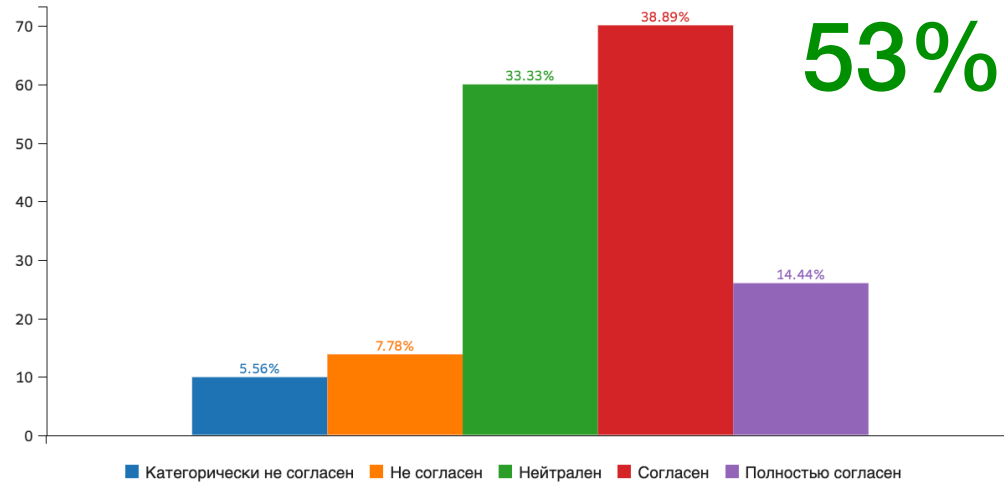
Более 50% разработчиков более удовлетворены своей работой и меньше устают, более 1/3 получает возможность концентрации на более важных задачах



# GigaCode оптимизирует рабочее время разработчиков

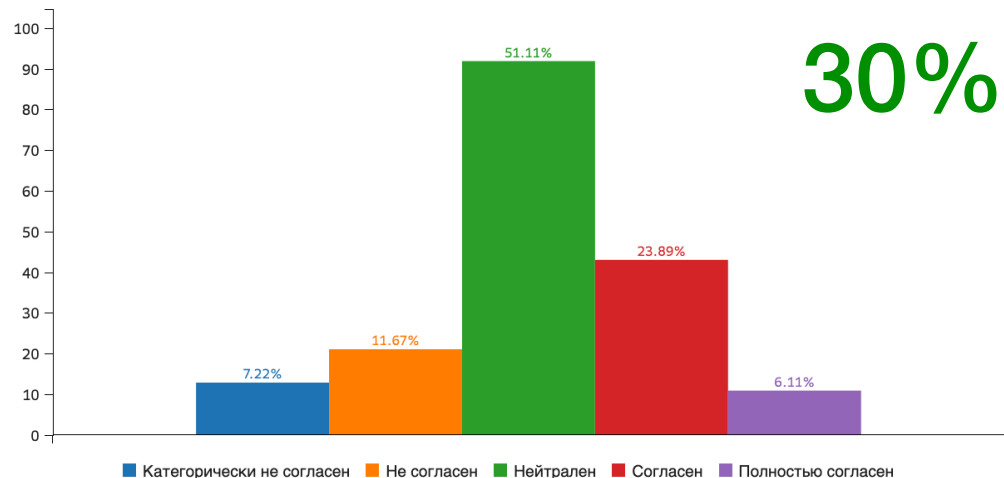
8

Трачу меньше времени на поиск информации и примеров кода

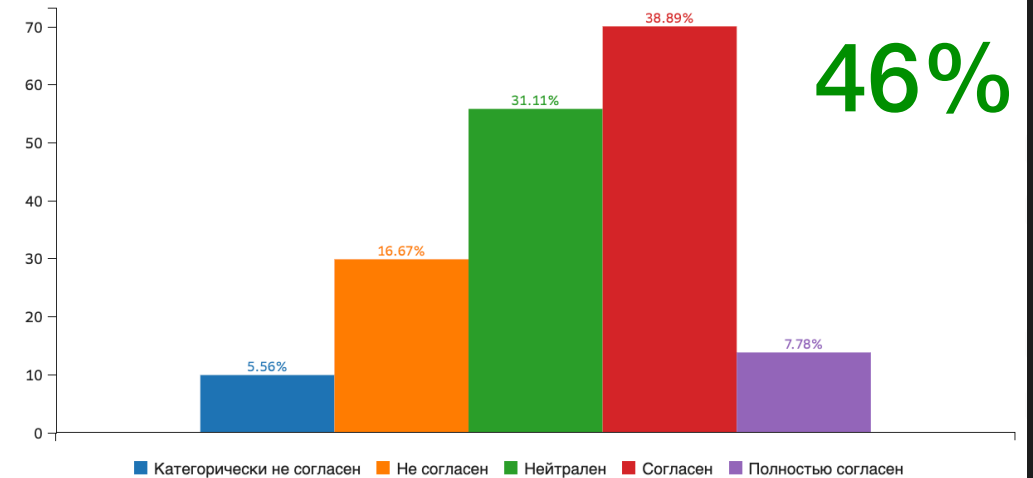


а также повышает их  
квалификацию

Меньше переключаюсь между задачами и больше нахожусь в состоянии «потока»



Учусь на подсказках, которые предлагает мне GigaCode



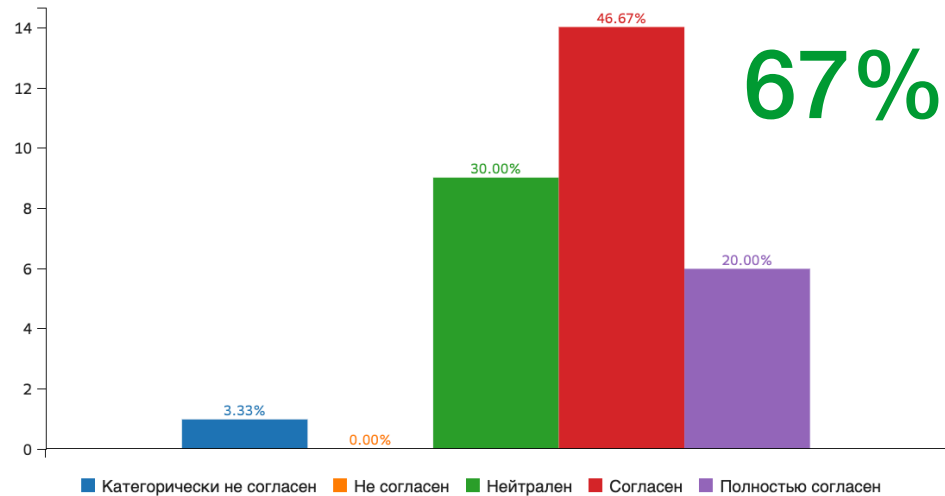


# GigaCode значительно повышает эффективность и помогает начинающим разработчикам

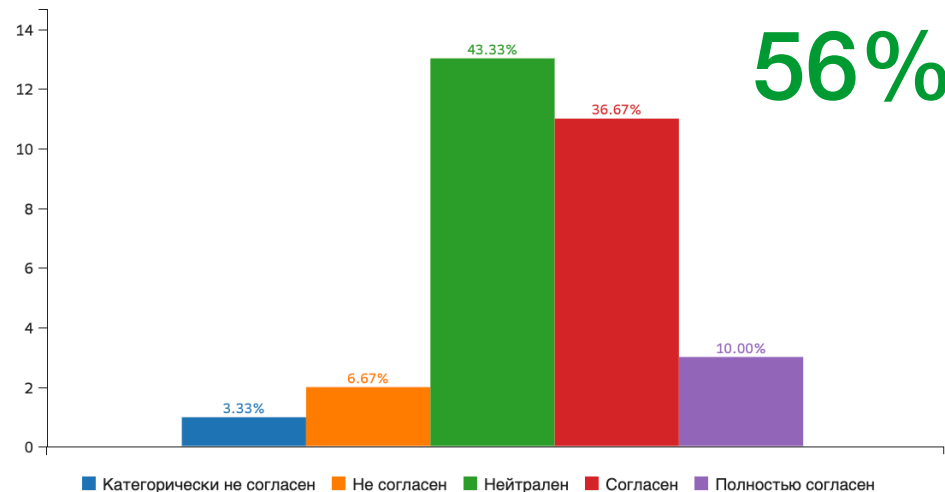
9

## Junior

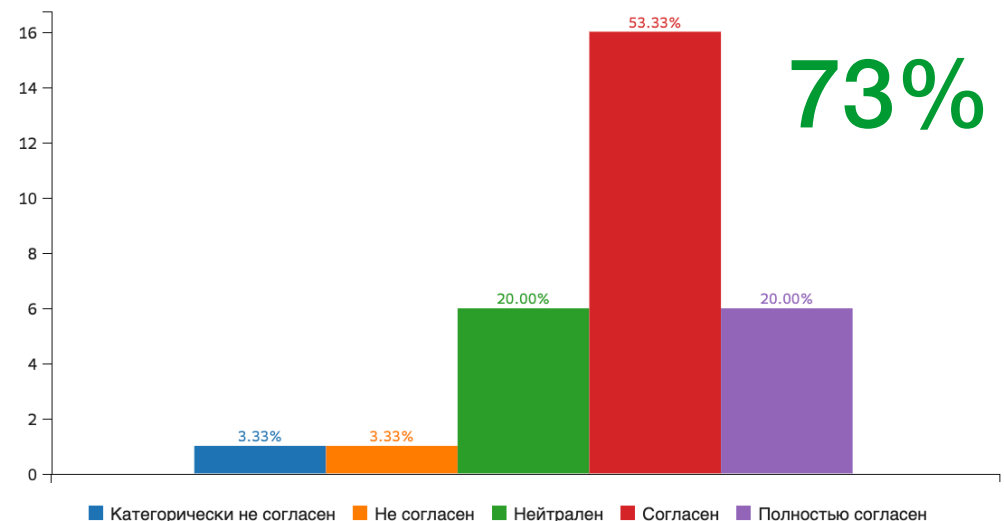
Трачу меньше времени на поиск информации и примеров кода



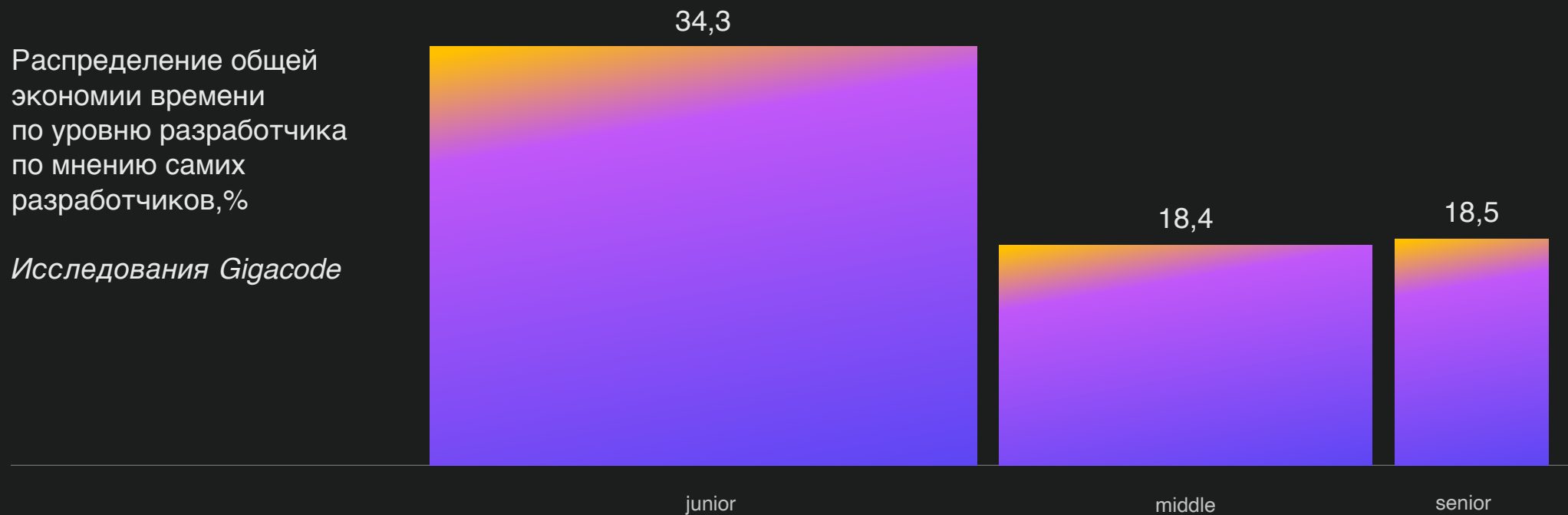
Меньше переключаюсь между задачами и больше нахожусь в состоянии «потока»



Учусь на подсказках, которые предлагает мне GigaCode



# Эффективность GigaCode среди разработчиков различных уровней



# Точки входа в ассистента GigaCode

## Inline Code Assistant

*Автодополнение, продолжающее  
кода после курсора.*

- 
- ✓ Реализует как однострочные, так и мультилайн подсказки
  - ✓ Подсказывает в месте написания кода, уменьшая количество рутины

## Chat with GigaCode

*Интеллектуальный чат, Находящийся в  
отдельной вкладке Chat в IDE.*

- 
- ✓ Помогает быстро находить ответы на вопросы, не выходя из среды IDE
  - ✓ Благодаря чату можно освоить новые сферы и изучить новые технологии

# Автодополнение кода: inline suggestions, multiline, lookup

## Inline Code Assistant

Интегрированное в редактор кода  
автодополнение, подсказывающее  
вероятные продолжения строки исходя  
из контекста текущего файла  
и проекта сразу после курсора

Однострочная  
подсказка

```
1  const http = require('http')
2  const fs = require('fs')
3  const port = 8080
4
5  const server = http.createServer((request, response) => {
6    res.writeHead(200, {'Content-Type': 'text/html'})
7    write
8    wrap (HTMLTextAreaElement, DOM)
```

Многострочная  
подсказка

```
private static class MyWelcomeHandler implements HttpHandler {
    @Override
    public void handle(HttpExchange httpExchange) throws IOException {
        httpExchange.sendResponseHeaders(200, "Welcome to Sber!".length());
        OutputStream os = httpExchange.getResponseBody();
        os.write("Welcome to Sber!".getBytes());
        os.close();
    }
}
```

Comments  
to code

```
1
2  # write a function that generates a random password with a specific length
3  def random_password(length):
4      import random, string
      password = ""
      for i in range(length):
          password += random.choice(string.ascii_letters + string.digits)
      return password
```

MacBook Pro

## Chat with GigaCode

Чат с ассистентом, обладающий как стандартным набором возможностей, так и работой с текущим окном кода. Помимо этого имеет набор коротких команд: объяснение кода, генерация документации и тестов

GIGACODE

CODECHAT INLINE

Clear chat

Me

Отрефактори следующий код, улучшив его эффективность и читаемость с использованием современных возможностей JavaScript

GigaCode

Вот рефакторинг кода с использованием современных возможностей JavaScript, таких как async/await и Promise.all:

```
async function fetchDataAndProcess(urls) {
  try {
    const results = await Promise.all(urls.map(u
    console.log('All data fetched and processed'
    return results;
  } catch (error) {
    console.error('Error fetching data:', error)
  }
}
```

В этом коде мы используем Promise.all для одновременного выполнения всех запросов fetch. Это значительно ускоряет процесс получения данных, особенно если количество URL-адресов велико. Кроме того, мы используем async/await для более удобного и читаемого кода.

References (1)

task3.js:3-18

+ New chat

Ask GigaCode a question or start with / for command ...

GIGACODE

CODECHAT INLINE

Clear all chats

Преобразуй следующий код с использованием M...

Здесь представлен эквивалентный код с использ...

Преобразуй следующий JavaScript код в TypeScri...

Вот как можно преобразовать данный JavaScript ...

Отрефактори следующий код, улучшив его эффек...

Вот рефакторинг кода с использованием совреме...

# Короткие команды: explain, explain step by step, doc, test

Пример  
использования

```
GigaCode: explain | explain step by step | doc | test
1 function transformData(input) {}
2   if (Array.isArray(input)) {
3     return input.map(item => item.trim().toUpperCase())
4   } else if (typeof input === 'string') {
5     return input.trim().toUpperCase()
6   } else {
7     throw new Error('Invalid input type')
8   }
9 }
```

## TEST LIBRARIES

```
/Jest
/Mocha
/Jasmine
/Ava
/Tape
```

+ New chat

/test prog\_lang:js;lang:eng;

@ Me

/explain\_stepbystep lang:ru;

GigaCode

1. Функция fetchDataAndProcess(urls) принимает массив URL в качестве аргумента.
2. Создается пустой массив results, в который будут добавляться результаты запросов.
3. Начинается цикл for, который проходит по каждому URL в массиве urls.
4. Для каждого URL вызывается функция fetch, которая отправляет HTTP-запрос на указанный URL.
5. Полученный ответ от сервера обрабатывается с помощью метода .then(response => response.json()). Этот метод преобразует ответ в формат JSON.
6. Полученные данные (data) добавляются в массив results с помощью метода .push(data).
7. Если длина массива results равна длине массива urls, то выводится сообщение 'All data fetched and processed' в консоль. Это означает, что все данные были получены и обработаны.
8. Если во время выполнения кода произошла ошибка, она обрабатывается с помощью метода .catch(error => console.error('Error fetching data:', error)). Эта ошибка выводится в консоль.
9. После обработки всех URL функция fetchDataAndProcess(urls) завершает свою работу.

References (1)

task3.js:3-18

## Подведем итоги интеграции ИИ в процессы разработки

Снижение  
времени на  
рутинные задачи

Общее повышение  
скорости разработки

Снижение кривой  
обучения для новых  
разработчиков

Повышение  
удовлетворенности  
при работе



<https://gigacode.ru>