Introduction

SWE233: Intelligent User Interfaces

Daye Nam Fall 2025

Introductions

Daye Nam

B.S. Yonsei University 2016

M.S. University of Southern California 2018

Ph.D. Carnegie Mellon University 2024

Research Scientist at Google 2024-2025

Assistant Professor at UC Irvine

You?

Your (preferred) name Why are you taking this course? A fun fact

Activity

Intro survey

https://forms.gle/EJJq5qGcSzhPSZV96



Intelligent User Interfaces

Intelligent User Interfaces

User interface

Everything the user encounters when using a product (e.g., a particular machine, device, computer program, or system)

Intelligent user interface

User interface that is intelligent

Intelligent user interfaces can

complement human capabilities by identifying and augmenting

shortcomings in what people can do;

be integrated into the tools that people and organizations already use to structure and carry out their work

Questions we will ask

How can interaction be made clearer and more efficient?
How can interfaces offer better support for their users' plans and goals?
How can informatino be presented more clearly and effectively?
How can the design and implementation of good interfaces be made easier?

Questions that have been asked for ~50 years

How can interaction be made clearer and more efficient? How can interfaces offer better support for their users' plans and goals? How can informatino be presented more clearly and effectively? How can the design and implementation of good interfaces be made easier?



• "... It addresses the same questions that have driven research in the gereral area of user interfaces for the past ten years:"

ELIZA



Created by Joseph Weizenbaum in 1964

One of the first natural language interfaces

Knowledge Navigator



Apple concept video from 1987

Created by Hugh Dubberly et al., for a keynote speech of CEO John Sculley

Visily



Google Glass (old version)



How can interaction be made clearer and more efficient?

ELIZA, by supporting a natural language-based interaction and mimizing the necessary domain knowledge (e.g., programming) and cognitive load

How can interaction be made clearer and more efficient?

ELIZA, by supporting a natural language-based interaction and mimizing the necessary domain knowledge (e.g., programming) and cognitive load

How can interfaces offer better support for their users' plans and goals?

Knowledge navigator, by acting as a proactive, intelligent agent that understands high-level, conversational intent and automates the complex workflows required to achieve those goals.

How can interaction be made clearer and more efficient?

ELIZA, by supporting a natural language-based interaction and mimizing the necessary domain knowledge (e.g., programming) and cognitive load

How can interfaces offer better support for their users' plans and goals?

Knowledge navigator, by acting as a proactive, intelligent agent that understands high-level, conversational intent and automates the complex workflows required to achieve those goals.

How can information be presented more clearly and effectively?

Google glass, by integrating information directly into the user's field of view and reducing the cognitive load required to switch between a physical task and a separate screen

How can interaction be made clearer and more efficient?

ELIZA, by supporting a natural language-based interaction and mimizing the necessary domain knowledge (e.g., programming) and cognitive load

How can interfaces offer better support for their users' plans and goals?

Knowledge navigator, by acting as a proactive, intelligent agent that understands high-level, conversational intent and automates the complex workflows required to achieve those goals.

How can information be presented more clearly and effectively?

Google glass, by integrating information directly into the user's field of view and reducing the cognitive load required to switch between a physical task and a separate screen

How can the design and implementation of good interfaces be made easier?

Visily, by automatically convert a static image into a fully editable mockup and reducing the user loads

Course Themes

- Foundations of Human-Computer Interaction and UI Design Principles
- User Modeling and Personalization Techniques
- Adaptive and Context-Aware Interfaces
- Conversational AI and Natural Language Interfaces
- Agentic Interfaces and Multi-Step Task Delegation
- Multi-Modal Interfaces (Voice, Gesture, AR/VR)
- Programming by Demonstration and Example-Based Systems
- Research Methods for HCI (Surveys, Interviews, Qualitative Analysis)
- Evaluation Methods for Intelligent UI (A/B Testing, Longitudinal Studies, Wizard of Oz)
- IDEs and Al-Powered Programming Tools
- Debugging and Search Interfaces
- Data Science Workflows and Computational Notebooks
- Explainable AI (XAI) in User Interfaces
- Trust, Transparency, and User Control in Intelligent Systems

Course logistics

Evaluation

```
80% Research Project
    5% Problem Identification
    25% Project Proposal
         Report: 15%
         Presentation: 10%
    15% Prototype Design
    35% Final Report
         Report: 15%
         Presentation: 10%
         Holistic evaluation: 10%
10% Paper Presentation
10% In-Class Participation
[Optional] 10% Prototype Implementation
```

Research Project

A quarter-long project, where you design and prototype an intelligent user interface.

Expected outcome:

A CHI/UIST/ICSE/FSE/ACL/Neurips... paper except for the results

Abstract

Introduction

Literature review

Methods

Evaluation

SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering

John Yang* Cu

Language model (LM)

tasks in digital envir

applications such as it software engineering users with their own interfaces to the softw

performance of langua SWE-agent: a system

solve software engine (ACI) significantly on

entire repositories, and

with a pass@1 rate of

state-of-the-art achiev

how the design of the

Princeton

"What It Wants Me To Say": Bridging the Abstraction Gap Between End-User Programmers and Code-Generating Large Language Models

Michael Xievang Liu' Microsoft Research, Carnerie

Advait Sarkar Microsoft Research Carina Negreanu Microsoft Research







Micro

Code-generating la

Recent work has demonstrated back [39]. However, applying a unexplored. To solve programs tions, such as the Linux shell or programming tasks such as soft plications like VSCode with po (HCI) studies on the efficacy of could similarly benefit from be

1 Introduction



Figure 1: SWE-agent is an LN (ACI), which includes the comm Tigual contribution. Comunic

Data, code, and leaderboard

38th Conference on Neural Inform

ABSTRACT

code. However, on uralistic utterances non-expert end-use of abstraction matel context of data and the user's natural la generator, executes grounded abstraction by translating the co ralistic atterance. Is

> We find that the or standing of the scor and the kind of lang CCS CONCEP · Human-centered

we compare groun

ternative based on a

Interactive systems KEYWORDS Natural Language 1

tion, Large Languay Equal contribution

Permission to make dig classroom use is granted for profit or commercial on the first page. Copyri author(s) must be honore republish, to post on serv and/or a fee. Recogst on CNT '28, April 29-28, 282 © 2023 Copyright held b ACM ISBN 978-1-4803-9

https://doi.org/10.1145/31

Andrew Macyean Dave Nam Google, Inc.

vasilescu@cmu.edu

Carnegie Mellon University U.S.A. dayen@cs.cmu.edu

Vincent Hellendoorn Carnegie Mellon University U.S.A. vhellend@andrew.cmu.edu

U.S.A. amacvean@google.com Bogdan Vasilescu Brad Myers Carnegie Mellon University U.S.A.

Using an LLM to Help With Code Understanding

Understanding code is challenging, especially when working in new and complex development environments. Code comments and focumentation can help, but are typically scarce or hard to navigate Large language models (LLMs) are revolutionizing the process of writing code. Can they do the same for helping understand it? In this study, we provide a first investigation of an LLM-based con renational UI built directly in the IDE that is geared towards code understanding, Our IDE plugin queries OpenAl's GPT-3.5-turbo model with four high-level requests without the user having to write explicit prompts: to explain a highlighted section of code, provide details of API calls used in the code, explain key domainpecific terms, and provide usage examples for an API. The plugin also allows for open ended promots, which are automatically contextualized to the LLM with the program being edited. We evaluate this system in a user study with 32 participants, which confirms that using our placin can aid task completion more than web search. We additionally provide a thorough analysis of the ways developers use, and perceive the usefulness of, our system, among others finding that the usage and benefits differ between students and pocfessionals. We conclude that in-IDE promot-less interaction with

ACM Reference Formati

Dave Nun, Andrew Marvern, Vincent Hellendroen, Booden Varilescu. and Brad Myers. 2024. Using an LLM to Help With Code Understanding. In 2839 TEE/ACM 460) Interactional Conference on Saftware Regimenting (ICSS) 260, April 14-20, 2024, Linbox, Partagol, ACNI, New York, NY, USA, 23 pages https://doi.org/10.1145/3597513.3659185

LLMs is a promising future direction for tool builders.

1 INTRODUCTION

Building and maintaining software systems requires a deep understanding of a codebase. Consequently, developers spend a sixnificant amount of time searching and foraging for the information they need and organizing and digesting the information they



ECSE 24. April 14-26. 2024. Linkon, Portunal

D 2024 Copyright held by the owner/confort) ACM 2020 979-8-407-4227-4/24/94 https://doi.org/10.1145/2597507.3659187

Carnegie Mellon University U.S.A. bam@es.emu.edu

find 130, 31, 34, 44, 48, 571. Understanding code, however, is a challenging task; developers need to assimilate a large amount of information about the semantics of the code, the intricacies of the APIs used, and the relevant domain-specific concepts. Such information is often scattered across multiple sources, making it challenging for developers, especially povices or those working with unfamiliar APIs, to locate what they need. Furthermore, much of the pelevant information is inadequately documented or spread across different

formats and mediums, where it often becomes outdated. With the growing popularity of large language model (LLM) based code generation tools [26, 54, 67], the need for information support for code understanding is acquably growing even higher These tools can generate code automatically, even for developers with limited coding skills or domain knowledge. This convenience comes at a cost, however - developers may receive code they don't understand [24, 79]. Indeed, early research on LLM code generation tools has found that developers have a harder time debugging code generated by the LLM and easily get frustrated [4), 71].

Fortunately, LLMs also provide an opportunity in this space namely by offering on-demand generation-based information support for developers faced with unfamiliar code. Compared to general web search queries [74], LLM prompts can allow developers to provide more context, which can enable them to receive information that note medically aliens with their specific needs, notentially reducing the time spent on silking through the information obtained from the web to sait their particular requirements. Developers have indeed taken to web-hosted conversational LLM tools, such as ChatGPT for programming support en masse, but this setup requires them to both context switch and copy the relevant context from their IDEs into the chat system for support.

To explore the potential for generation-based information support directly in the developer's programming environment, we developed a prototype in IDE LLM information support tool, GILT (Generation-based Information-support with LLM Technology) GILT is capable of generating on-demand information while considering the user's local code context, which we incorporate into the prompts provided to the LLM behind the scenes. This way, we also introduce a novel interaction method with the LLM, prompt-less interaction. This option aims to alleviate the cognitive load associated with writing prompts, particularly for developers who possess imited domain or programming knowledge.

As there is still little knowledge about how to best use an LLM for information support (as opposed to just code generation), we evaluate the effectiveness of our prototype tool in an evaluratory

Project (Tentative timeline)

Oct 8: Problem Identification Identify a problem for intelligent UI solution

Oct 27: Project Proposal Literature review and proposed solution
Oct 20&22: Proposal Presentation Present your proposal for feedback

Nov 17: Prototype Design System architecture and interface design

Dec 10: Final Report Complete evaluation methodology

Dec 1 & 3: Final Presentation Showcase your complete project

Paper Presentation & Discussion

We will read and discuss systems with intelligent user interfaces, mainly within the software engineering domain

Presentaion

Each student will present at least one paper over the quarter.

As presenters, you will do a presentation of the paper and connect the paper to many aspects we cover in the first part of the quarter.

Paper selection

You will bid for a paper you want to present in the next class.

Bid for papers that are close to your potential project topic.

Course infrastructures and logistics

Infrastructure/source of truth

Course website: schedule, slides, syllabus, office hours

Canvas: homework, grades, other material

Slack: communication and collaboration (will send you today)

Logistics

Lecture in-person only

You are required to attend presentation sessions (proposal/final)

Office hours: Monday 4:00 pm - 5:00 pm, but flexible