# *[MetaGPT] Using LLMs to help people engage in design thinking*

***Daniel Flynn***

***2469113F***

## Proposal

### Motivation

Design thinking problems are simultaneously complex and ambiguous decision-based tasks. While classical software engineering is effective at creating time efficient solutions to complex problems, it by nature requires a rigorous procedure to follow, including a static method to making decisions. Large Language Models provide potential solution to the clash of requirements between software engineering and design thinking, as it mitigates the need for precision in normal software, and when asked to make decisions its response will be a complex, probability-based predictions not reliant on a method provided by the coder. An example of this can be seen with understanding contextual background: say we want to give a program a page of human generated text as background information to a task. To write a deterministic program to analyse the text, we would have to program some static rules to incorporate understanding of the English Language, and then some more static rules to extract what in the text is important. Not only is the number of potential variables not realistic to deal with, but the rules of the English language are also ambiguous to begin with and no set of decision rules will consistently capture what is *important* information. However, LLMS provide the best solution to these issues we have; inferences on the model capture some sense of *understanding*, and predictions capture *decisions.* However, LLMs present a different issue: their most common *prompt and response* form is simple, and we want to use them for complex tasks. MetaGPT provides a solution to scaling LLMs up to complex and multifaceted tasks, by providing a framework for coordinating multiple LLM agents, each specialised to complex a specific simple subtask. When these subtasks are sequenced, the overall process can become a solution to complex tasks. The motivation of this project is to apply this principle to design thinking, by taking design thinking Standard Operating Procedures and decomposing them into subtasks each agent can complete.

### Aims

I've chosen to specific this project to the innovation phase of design thinking, specifically product discovery. This means I want to be able to provide the program with a background, and it can come up with original and viable product ideas. It should follow established design thinking Standard Operating Procedures, creating a standardised output for each subtask mirroring what a real design team would produce. To do this the program will follow the Double Diamond design pattern. While what is a *good* idea is hard to quantify, the ideas generated will be tested through comparison with a design student. This will be done through a blind test, where the program and student will both be given the same background and time to come up with product ideas. An independent design student/professional will then be presented with the ideas, and their feedback on which is

better will be the main metric on which success is evaluated. Ideally, a successful project will outperform the student.

## Progress

- Background research on design thinking
- Background research on MetaGPT
- Successfully created a working Double Diamond process for generating product ideas when given a topic, background, timeframe, and budget
- Process currently consists of the following subtasks:
    - Create Personas for customers who would be affected by the topic/background
    - Analyses specific needs of each customer
    - Identifies what are the fundamental needs of each customer
    - Creates product ideas based off meeting these needs
    - Analyses and scores ideas based of feasibility
    - Analyses and scores ideas based on market viability
- Each subtask is implemented in the MetaGPT framework, with agents given specific roles and actions

## Problems and risks

### Problems

- When first starting work on the project, there was no documentation for MetaGPT meaning a lot of time had to be spent reading the codebase to understand how the framework worked
- Hard to find academic research on design thinking, meaning a lot of uncertainty in what I should be coding as uncertain on correct SOPs to follow
- MetaGPT itself is not actually a finished product -> only on version 0.3 right now and had to wait for an update for some features I needed to be added
- Lots of hard to debug issues from working with an unfamiliar codebase, for example recognising and dealing with issues from running out of completion tokens on API responses.
- Poor quality of responses from GPT 3.5, but GPT 4 is too costly
- Program long time becoming too long to easily debug

## Risks

- API costs slowly rising
    - Mitigation: Create “checkpoints” in the program that allow me to save responses from previous runs and use them to test specific subtasks
    - Mitigation: Stick with GPT 3.5 until testing to keep costs minimal
    - Mitigation: Some sort of funding would still be ideal
- How to evaluate -> what is a “good” design idea/output from the program
    - Mitigation: Use blind test as described in Aim
    - Still concerned
- Still struggling to identify suitable Standard Operating Procedures with a justifiable background and are suitable for MetaGPT
    - Mitigation: Recently found Googles Design Sprint Kit (https://designsprintkit.withgoogle.com/), going to base procedures off the methods outlined here

## Plan

**Semester 2:**

- **Week 1-2:** Finalise initial SOP (implement each subtask) based off Google Design Sprint Kit
    - Deliverable: Working SOP -> put in a background and topic, get out design ideas
- **Week 3:** Go over initial SOP with design student and refine process
    - Deliverable: Refined SOPs
- **Week 4-5:** Create graphic outputs rather than text for subtasks
    - Deliverable: Graphic outputs from program
- **Week 6:** Plan evaluation, contact design students, come up with other metrics if possible
    - Deliverable: Evaluation plan
- **Week 7:** Polish software, if possible give it a basic UI, fix any outstanding issues
    - Deliverable: Finished software
- **Week 7-8:** Run evaluation tests
- **Week 7-10:** Begin write up
    - Deliverable: First draft submitted 2 weeks before final submission date