Modeling First: Rethinking Undergraduate Operations Management with AI

Andrew Daw, Vishal Gupta, and Paat Rusmevichientong
Dept. of Data Sciences and Operations, University of Southern California Marshall School of Business, Los Angeles, CA

As part of an institutional initiative to differentiate core undergraduate business classes, we have recently launched a new operations management (OM) course focused on modeling. Whereas the Traditional Operations Management (TOM) course reviews classical models in inventory, queueing, and supply chains, the new Operations Modeling with AI (OMAI) course centers on the *practice* of mathematical modeling, particularly optimization and simulation. A distinctive element of OMAI is the integration of generative AI–based pair-programming assistants through what we call Incremental Prompting, which prioritizes learning essential modeling skills—abstracting real situations and expressing them with mathematical precision—rather than programming syntax. We evaluated the course through an end-of-term self-assessment survey across both offerings (N = 250; TOM = 156, OMAI = 94). Results indicate that OMAI students reported greater comfort and confidence in understanding, valuing, and applying model-based decision-making. Interestingly, these findings become more pronounced after controlling for prior coding experience, suggesting that OMAI benefits both technically experienced and less experienced learners. In survey responses, students credited the pair-programming framework with their increased comfort and understanding of modeling. Taken together, our findings demonstrate how generative AI can be leveraged in OM and OR education to reduce technical barriers and sharpen focus on core modeling skills.

Key words: Undergraduate business education, Incremental Prompting, AI pair-programming, optimization and simulation modeling

1. Introduction

In the field of operations research (OR), there has long been a lingering sentiment that "OR has a PR problem." This concern has manifested in open reflections on the field's future (Kan 1989) and drift (Corbett and Van Wassenhove 1993), as well as in organizationally funded campaigns to market the profession, such as the clever "Science of Better" tagline (Sodhi and Tang 2008) and the recent emphasis on analytics (Liberatore and Luo 2010, Mortenson et al. 2015, Gorman 2021). Rarely have these concerns stemmed from the merits of the field itself. Rather, as the very existence of marketing campaigns suggests, there is a simple but essential need for more people to *learn about OR*.

Traditionally, and predominantly in engineering and mathematics programs, the approach to teaching OR has focused on the combination of models and methods (e.g., Jensen and Bard 2002). While methods are certainly important and intellectually engaging, it seems likely that, between

the two, models are the gateway to learning the discipline. Both a skillset and a mindset, modeling teaches students how to reason quantitatively about complex real-world problems in order to solve them. As the "simple models for complex realities" mantra suggests (e.g., Cai et al. 2019), this perspective provides a valuable toolkit across contexts.

By comparison, in operations management (OM) courses in business schools, the typical focus is neither on models nor on methods. Instead, curricula stress common tradeoffs and takeaways, largely by presenting the *results* of classical OR/OM models—the economic order quantity formula, a G/G/c waiting time approximation, etc. While this may be an effective strategy for distilling managerial insights for future business leaders, it does not teach the core skill of modeling, since students are given the model "upfront." Moreover, because these populations often exhibit highly varied mathematical preparation (and interest), such courses rarely delve deeply into the underlying OR methods. Thus, traditional OM courses ultimately leave students with a poor grasp of what OR and OM practitioners actually do. Moreover, the prevailing course's perspective jumps straight to the long-run and overlooks the critical initial phases. Particularly in the context of undergraduate business education, students do not graduate and simply immediately assume senior leadership roles. On the contrary, in an increasingly data-driven world, more and more entry-level roles have become increasingly tangential, if not outright reliant, on quantitatively reasoned decision-making.

We claim that AI presents an opportunity to revisit this pedagogical challenge in undergraduate OM courses. As part of a larger institutional initiative to differentiate undergraduate business courses, we introduced a version of our core OM course *specifically* focused on the skill of modeling. A key element of our course design is leveraging generative-AI-based pair-programming tools (e.g., GitHub Copilot) through a prompting strategy we call *Incremental Prompting* (see Section 1.1). Importantly, Incremental Prompting circumvents what we see as the major hurdle in teaching modeling to undergraduates with varied technical backgrounds: for students to appreciate the value and impact of a model on practice, it must be implemented, and to implement it, they need to code it in some software. But learning syntax or the idiosyncrasies of a particular platform is an orthogonal (and often challenging) task that distracts from the primary learning objective of modeling. Through our Incremental Prompting approach, we move the pedagogical focus back onto modeling and outsource syntactical issues to the AI.

Evidence from the early success of our course suggests that undergraduate business students—both with and without prior technical backgrounds—report greater comfort with their ability to model and better understanding of the value of model-based decision-making in the new OMAI offering.

1.1. Our Approach to AI in the Classroom: Incremental Prompting, Not One-Shot Prompting

A key component of our new modeling-focused OM course is what we call "Incremental Prompting" for generative AI. Unlike the common approach of posing a single question and receiving a complete solution from a chatbot, Incremental Prompting breaks the task into a sequence of smaller, more specific prompts. This granularity serves a dual purpose in our course: improving output accuracy and reinforcing our pedagogical objective.

First, highly specific prompts for short code fragments reduce opportunities for error in the AI-generated response, and the brevity of the output makes mistakes easier to detect. Second, and *far* more importantly, Incremental Prompting requires students to think through the primitive "elements" of a model. These elements might be a set of variables representing production, or a family of constraints capturing flow conservation. They do not necessarily correspond to a single line of code, but rather to a distinct semantic idea in the model. Experienced modelers often reason about large models in these kinds of "semantic chunks." By encouraging students to prompt syntactically line-by-line, we are implicitly teaching them to decompose and conceptualize models in this way. We see this as a key learning objective and a foundational skill for modeling.

We stress that Incremental Prompting is distinct from the more common strategy of providing an entire natural language description to a chatbot and requesting a full mathematical formulation or implementation. Such an approach is often error-prone (see Sec. 2) and, more importantly, tends to avoid or even replace the act of modeling. By contrast, Incremental Prompting requires students to engage directly in the modeling process to construct the relevant, line-by-line prompts themselves. In our courses, we have students first write out mathematical formulations of models with "paper and pencil" before beginning to code. In this sense, we view Incremental Prompting as a pedagogical tool teaching modeling, distinct from approaches that seek to automatically generate optimization formulations to replace modelers.

Figure 1 shows a pair of screenshots illustrating the Incremental Prompting idea through the implementation of an optimization model using the Gurobipy library in Python. Both snippets attempt to implement a budget constraint as the next increment of a model after its decision variables have been defined above. In both cases, the implementation is assisted by GitHub Copilot, a generative-AI pair-programming tool within the course's coding platform (which we will detail further in Section 3). This assistant is more like an auto-complete than a chatbot: prompting with Copilot is done through writing comments (which take place on lines with "#" in Python), and the AI-generated responses then come in the form of coding suggestions output underneath the comment.

The contrast between these two example screenshots highlights the two purposes of Incremental Prompting. In the upper screenshot of Figure 1, the prompting comment is described in plain

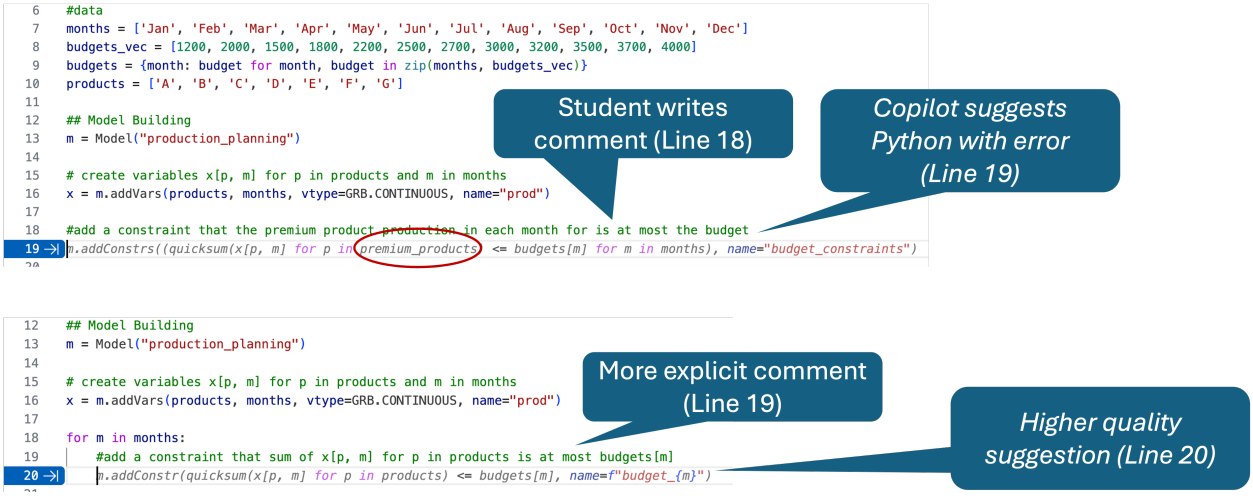

Figure 1 *Incremental Prompting.* Students write short, explicit comments describing self-contained modeling steps. They proceed *incrementally* to build up an executable model. In the first example, a very generic prompt leads to a hallucination - referencing an undefined variable "premium_products." In the second example, the explicit, mathematical form of the prompt elicits a higher quality suggestion.

language and not mathematically precise: the idea of the constraint is described rather than the constraint itself. By comparison, in the lower screenshot, the prompting comment actually describes the constraint in terms of the data and decision variables, and it even does so within a for loop to further reduce input uncertainty. In result, the loose language of the upper screenshot produces an error, with the AI-generated code using a hallucinated data variable in the suggested constraint, whereas, in the lower screenshot, the specific mathematical language correctly yields the desired constraint.

Notice that the resulting Incremental Prompting workflow is to interpret a problem, develop a model, and then implement and solve it by describing it incrementally and precisely. This approach keeps the focus and primary responsibility of students on *modeling*, i.e. translating reality to math. The goal is not to teach students to code, nor is it to teach them to prompt AI about a problem in one-shot fashion to receive a wholesale model. Hence, our approach to AI in this course keeps students' hands and eyes on the modeling concepts at the course's core.

1.2. Organization of Paper

In the remainder of this paper, we will provide further details on our approach to teaching modeling in undergraduate OM courses with AI, including the results of a modeling self-assessment survey of students from both the new course and the traditional OM offering at our institution. First, in Section 2, we survey related literature, including on generative AI use in optimization formulation and generative AI in education. In Section 3, we describe our new course in greater depth, including

content, delivery, and assessment, and we also provide background context for the traditional OM course. In Section 4, we analyze the results of the student survey, where we find that both students with strong technical backgrounds and those without prior technical experience seem to be well-served by the new course, with AI assistance reported as a key component of student success (with important caveats). Finally, in Section 5, we provide additional details of the course and reflections for possible improvements in future semesters. The full results of the survey are available in the appendix, as is the complete collection of survey questions.

2. Related Work

2.1. Using Generative AI to Formulate Mathematical Optimization Problems

We contrast our work with recent attempts to use generative AI to translate natural language descriptions of optimization problems into executable code or mathematical formulations. This task was the subject of the recent NL4Opt Competition (Ramamonjison et al. 2023). Various approaches have emerged for the problem. AhmadiTeshnizi et al. (2025) develops a system that takes in the natural language description of an optimization problem and attempts to formulate it, code it for a solver, call the solver, and debug its own errors. Xiao et al. (2024) proposes an alternate "chain of experts" strategy that links together different LLMs with prescribed subtasks to achieve a similar goal. Other authors have looked at more specialized tasks. For example, Lawless et al. (2024) use natural language descriptions of constraints to formulate and solve a constraint programming problem for meeting scheduling. Importantly, all of these approaches are *not* aimed at students. They are meant to fully replace the need for an expert modeler, and are perhaps best suited to application contexts where established models exist, but are hard to tailor to an organization's specific needs.

By contrast, we are interested in teaching modeling as a skill and hence use a less "black-box" approach in our classroom. In the context of optimization, this involves i) developing a paper-and-pencil model formulation (similar to a traditional optimization classes) and interactively ii) writing very explicit code comments to Github Copilot to describe the variables and constraints while iii) reviewing the few lines of code generated by Copilot as it "translates" that comment. Steps ii) and iii) are done interactively and a single comment at a time, so that the student can see the model organically come together, as according to the Incremental Prompting framework given in Section 1.1. Overall, students must still think through a problem context to identify variables and constraints, themselves, and thus they engage in modeling in the same way a mature operations researcher might. We think this approach is critical to learning fundamental modeling skills and hence also design our assessments around it.

2.2. Generative AI in Higher Education

The use of AI, generally, and generative AI, more specifically, in education is an active subject of ongoing research with studies across educational levels investigating different aspects of education, from different stakeholder viewpoints, and with respect to different AI tools. An exhaustive summary is impossible. We focus our literature review primarily on the use of generative AI tools in higher education.

Yan et al. (2024) surveys different use cases of LLMs in education, focusing on practical obstacles to adoption and ethical issues around transparency, privacy, equality, and beneficence. They identified 53 use cases under 9 main categories: profiling/labelling, detection, grading, teaching support, prediction, knowledge representation, feedback, content generation, and recommendation. Our work – leveraging GitHub Copilot for code generation – falls under "content generation." Similarly, Belkina et al. (2025) surveys case studies, discussing the challenges of "ensuring that educational technologies are used effectively, ethically, and inclusively." Bien et al. (2024) is another notable survey centering on use cases specifically in the business (both undergraduate and graduate) curriculum.

Perhaps closer to our focus is the growing literature on whether generative AI tools improve student learning in higher education. Dong et al. (2025) provide a recent survey. They note that past research has a varied focus, some authors studying the effects of a particular tool on student achievement (as measured by a standardized assessment) and others on educational learning experiences. Moreover, this research is inconclusive: in some contexts, AI tools improve student achievement, while in others they have little or even negative effect. These past works span educational contexts from nurse training (Chang et al. 2022) to massive open online courses (MOOCs) for elementary school students (Bachiri et al. 2023). Dong et al. (2025) consequently perform a meta-analysis of past work (spanning generative AI and other AI tools across multiple education levels) and overall find that the introduction of AI tools improves student achievement. Wang and Zhao (2024) and Zheng et al. (2023) also perform similar meta-analyses (again across all AI tools and educational contexts) finding positive effects of introducing AI tools on student achievement, although Zheng et al. (2023) surprisingly finds smaller effects on self-perception of learning and experience. We stress, however, there is far from a consensus on the benefits of generative AI tools. Bastani et al. (2025) conducts a large RCT with high school math students and argues that LLM tutors can hurt long-term learning when students can no longer access the tutor.

Our study differs from these previous works in an important respect: In most studies, AI tools are used to help assist in achieving the primary learning objective. For example, in Bastani et al. (2025), tutors aim to teach students mathematics, and, later, students are assessed via an exam on

mathematics. By contrast, in our work, pair programming is used to help students translate code comments into executable code, but our primary learning objective is not teaching students to code. Rather, it is teaching students how to model, and the relevant modeling steps are still performed by hand and assessed with "paper and pencil." (See Section 3 for further course details.) The use of pair-programming tools like Github Copilot is to develop code that augments understanding of the model and connects material to real-world applications. Thus, while one might plausibly argue that Copilot hinders students' learning of Python, it is less clear how it affects their learning of modeling skills.

In this respect, our use of pair-programming more closely resembles the use of software tools throughout STEM education. Recently, generative AI tools have been used in introductory statistics and data science courses (Bien and Mukherjee 2025, Bray 2025). Within OR, there is a long history of using spreadsheets (i.e., Microsoft Excel) for teaching both simulation (Evans 2000, Hill 2002, Eckstein and Riedmueller 2002) and optimization (Pachamanova 2006, Mason 2012, Martin 2010). Web-based tools for simulation (Dobson and Shumsky 2006, Snider and Balakrishnan 2013) and interactive games (Chen and Samroengraja 2000, Pasin and Giroux 2011, Griffin 2007) are also common, particularly in supply-chain. For optimization problems, specifically, authors have also advocated for various mathematical modeling languages (Mason 2013, Dunning et al. 2017, Bynum et al. 2020, Dunning et al. 2015, Lofberg 2004, Grant et al. 2006, Fourer et al. 1990, Chen and Xiong 2023). The plethora of such software tools and best practice articles for them is strong evidence that there is wide agreement across the field that these types of tools support and promote student learning.

We see our use of GitHub Copilot to translate code comments to executable Python code as philosophically similar to these existing use cases, but also note a few advantages. Relative to Excel, coding in Python allows one to more easily represent large and complex optimization problems and simulation contexts. For example, when using Excel's Solver to model a linear optimization problem, one is essentially forced to set up individual linear constraints in matrix form. It is difficult, if not impossible, to represent the iteration concepts so often encountered in real-world constraints: "for every arc in the network, inflow equals outflow" or "every job should be assigned to one worker." Often, this makes it difficult for students to appreciate this level of abstraction when thinking about families of "related" constraints in large-scale applications. By contrast, these iteration concepts are naturally described with for loops and iterable collections (e.g., Python dictionaries). And, while one could, in principle, model them using a proprietary language like AMPL or JMP, doing so requires students to learn the programming syntax of that language while they are simultaneously struggling to learn underlying optimization concepts. By leveraging generative AI's ability to parse natural language text, students can focus on concepts and avoid

syntax entirely. Finally, once students understand iteration over constraints and variables, they can easily model large, real-world scale optimization problems that can motivate and inspire them. Similar arguments can be made with respect to creating discrete event simulations. Indeed, writing such simulations using Crystal Ball or @Risk can be notoriously finicky, requiring a great deal of "spreadsheet planning" to layout data and computations that obscures the fundamental concepts in simulation, while working directly in a programming language like Python requires a detailed discussion of syntax. In these various respects, we see our use of GitHub Copilot with Incremental Prompting as a potential improvement on existing software tools supporting OR/OM education.

2.3. Student Achievement vs. Student Self Assessment

We conclude by noting that, unlike some previous work which measures student achievement via standardized testing, our primary survey endpoint is a self-assessment of learned modeling skills. Indeed, given the practical constraints of our teaching environment – two, tracked core classes with overlapping, but distinct course topics and a need for consistency across sections and semesters – administering an appropriate standardized test across both groups and incentivizing them to perform well was logistically impossible. Admittedly, our survey results might have little information about actual cognitive learning, as Falchikov and Boud (1989) has shown at best a weak correlation between self-assessment and actual skills, with a wide variability across settings and contexts. Worse, when a correlation does exist, the seminal work from Dunning & Kruger shows that less-skilled people tend to systematically overestimate their abilities, while those more-skilled tend to underestimate (Kruger and Dunning 1999). Finally, there is a known strong confounding between instructional methods and the relationship between self-assessment and ability. For example, Deslauriers et al. (2019) show that under active learning, students tend to learn more, but perceive that they have learned less because of the additional cognitive effort required.

All that said, we still believe that our survey endpoint is useful to educators. Sitzmann et al. (2010) shows that self-assessment is a good predictor of self-motivation and satisfaction, while Duckworth and Yeager (2015) shows that, as a measurement, it is sensitive to experiential aspects of learning (like belonging and confidence) in a way that traditional standardized testing is not. Chemers et al. (2001), Multon et al. (1991) show a link between self-efficacy – a student's belief in their own ability to successfully perform a task – and academic persistence and academic performance. In these respects, while fully recognizing a rigorous assessment of differences in student achievement is still needed, we do believe our survey results are informative for educators experimenting with this style of course in their classrooms.

3. Curricular Context and Overview of OM Courses at Our Institution

As framed in Section 1, the motivation behind this article's focal course creation is essentially thus: there is a large and untapped market of learners receptive to OR models (i.e., undergraduate business students), and nascent leaps in technology (i.e., generative-AI-based pair-programming tools) make the present the perfect moment to meet this demand. Even if these learners do not go on to be professional modelers or OR practitioners themselves, they will be the next generation of team members and leaders in the workforce, not to mention a sizable portion of the general public. In fact, according to the National Center for Education Statistics, business constitutes the largest single category in terms of bachelor's degrees granted, with 375,418 degrees awarded in business majors (18.6% of all bachelor's degrees nationwide) relative to 123,017 degrees (6.1%) conferred across all engineering disciplines (Irwin et al. 2023). Moreover, the data from Irwin et al. (2023) also shows that undergraduate business students are broadly representative and reflective of the overall population across demographics. Hence, a course that advances the teaching of modeling and quantitative reasoning within undergraduate business education can pay dividends to the understanding, support, and adoption of these methods across a variety of workplaces and dimensions of society, which has been a long-running goal for our field.

In this section, we provide an overview of undergraduate operations management education at our institution and context for the two core classes: Traditional Operations Management (TOM) and Operations Modeling with AI (OMAI). TOM is a long-standing core course required of all undergraduate business majors, typically taken in the sophomore or junior year. It is considered one of the more quantitatively demanding courses in the curriculum. OMAI, first launched in Fall 2023, is the focus of this paper. Both courses fulfill the same core requirement, share the same prerequisites, and students are free to choose between them. As previously noted, the two courses differ significantly in content, pedagogical orientation, and assessment design. Informally, TOM aims to educate future managers on the processes and tradeoffs commonly encountered in operations, whereas OMAI emphasizes modeling and model-based decision-making.

3.1. The Traditional Operations Management (TOM) Course

TOM is likely a familiar course to anyone who has taught OM in undergraduate business education. As a core course, it aims to provide future managers with a foundational understanding of the processes and tradeoffs that typify operational contexts. Its primary objective is not to develop students as modelers or technical specialists, but rather to equip them with an operational mindset: to understand how decisions about things like capacity, inventory, quality, and pricing affect organizational performance, broadly speaking. Designed for undergraduate business majors, TOM emphasizes breadth over depth, with a focus on conveying managerial insights.

At this introductory, overview level, the OM canon is well-represented in TOM. The subjects covered include process analysis (e.g., bottlenecks and throughput), waiting and service management (e.g., Little's Law, psychology of wait, and an average wait approximation formula), optimization (e.g., small LP's and IP's, Excel Solver, sensitivity analysis through Solver, and decision trees), revenue management (e.g., linear demand models, price differentiation), and inventory and supply chain (e.g., EOQ, Newsvendor, reorder point policies). Additional topics like forecasting may be included by instructor interest, as can interactive game sessions, such as the beer game (Chen and Samroengraja 2000). Like most core courses at our institution, TOM is delivered in-person twice a week using a primarily lecture-based format. Though the inclusion of any resources is up to instructor preference, the class typically draws on classic OM textbooks (e.g., Cachon and Terwiesch 2008).

Learning assessment in TOM is primarily exam-based, with quizzes and tests being primarily quantitative in nature, as based on application of the formulas introduced in the various topics covered in class. Although questions may introduce slight twists or modified scenarios, students are not asked to model a problem from scratch. As aligned with the managerial focus of the course, many examples and assessments are case-based.

3.2. The New Modeling-Focused Alternative: Operations Modeling with AI (OMAI)

Given the sheer size of and variety of interests within the population of undergraduate business students at our institution, there has been a recognition over the last few years that our core courses could benefit from product differentiation. Following a precedent set in statistics, OMAI was developed to offer an alternative path to operations education for students interested in getting their hands dirty with modeling, data, and analytical tools. Whereas TOM emphasizes operational reasoning, OMAI focuses on the technical formulation and computational solving of quantitative decision problems. However, by comparison to OR courses as found in engineering programs, OMAI does not include algorithmic specifics, and it certainly does not present proofs. Its central aim is to build students' capabilities as modelers and quantitative reasoners – individuals who can abstract real-world systems into formal mathematical representations, apply algorithmic tools to obtain solutions, and, critically, translate those solutions back to decisions in practice. At its core, OMAI is a class about mathematical modeling.

The course remains targeted at business majors and requires no prerequisites beyond those for TOM, but it presumes an interest in – or willingness to engage with – technical material. Students must choose to take either TOM or OMAI in order to satisfy the requirements of the degree. Though OMAI has grown in each of the four semesters in which it has existed so far, TOM is indisputably the larger course. In the most recent semester, Spring 2025, there were three sections

of OMAI and nine of TOM. OMAI sections are also smaller by design, capped at approximately 40 students each, by comparison to TOM, which has 70 seats per section.

OMAI covers two common types of OM models: optimization and simulation. A brief opening unit on process analysis introduces essential tradeoffs in resource allocation and throughput, but the bulk of the semester is split between the two modeling approaches. The optimization half of the semester covers LPs and IPs (specifically, mixed integer linear programs) with a range of complexity. Topics covered include tightness of constraints and sensitivity analysis, variable indexing and constraint enumeration for large scale problems, logical constraints in binary optimization models, and linearization of problems through the addition of new decision variables or constraints. By comparison to the Excel-based optimization models covered in TOM, a central theme in OMAI is the translation of operational situations into a general mathematical form, which often involves construction of scalable models that are both not well-handled by the rigidity of a spreadsheet and not likely to satisfy the limitations on problem size as set by the default Solver limitations. Then, in the second half of the semester, OMAI focuses on Monte Carlo simulation. With the course being in its relative infancy, the simulation modeling material has varied across semesters, but each iteration has begun with a review of probability distributions as models of randomness, an introduction of decision trees for an initial manner of making decisions under uncertainty, and a primer on the essentials of discrete event simulation. Then, in the advanced topics that follow, queueing models have been covered every semester, whereas Markov chains, stochastic optimization, and dynamic programming have each been among the rotation in special topics of instructor interest.

Like TOM, OMAI also meets twice-per-week, and class sessions are a mix of lecture content and interactive problem solving sessions. However, unlike TOM, it does not adhere to a single textbook. Instead, instructional materials are drawn from a range of sources, including business-oriented texts (e.g. Bertsimas and Freund 2004) and methodologically focused references (e.g. Rardin 1998).

Instead, the most essential resource of OMAI has been the introduction of pair-programming assistants, such as GitHub Copilot. These generative-AI tools are based upon the same technology as well-known large language models like ChatGPT, but they have a crucial difference in interface. Rather than functioning like a chatbot, a pair-programming assistant is more like an auto-complete. It operates inside of an integrated development environment (IDE), such as Visual Studio (VS) or VS Code, rather than within its own interface. Similarly, instead of producing a full response to a prompt, it offers a single line (or, in some cases, a small number of lines) of code based on a comment. That is, a modeler using a pair-programming assistant to solve an optimization model would not submit the full model or problem scenario and expect a full implementation of the formulation in one fell swoop. Instead, as aligned with the Incremental Prompting philosophy,

they would enter the decision variables, objective function, and constraints in line-by-line fashion, successively describing each model element in plain (but mathematically precise) language and receiving an output line that translates that math to code.

As we outlined for the Incremental Prompting philosophy in Section 1.1, we believe this granular approach to model implementation with generative AI assistance helps new learners avoid pitfalls from LLM hallucination or other AI errors. Moreover, and perhaps even more importantly, the line-by-line interactions in pair-programming keeps the pedagogical focus on modeling. In this arrangement, the first-order task a student faces on any problem is to model. Once they fully conceive the model, they can then relay it piece-by-piece to the pair-programming assistant in order to implement and solve it.

Critically for the undergraduate business student population, this approach does not require any prior coding experience. In OMAI, we teach students how to use pair-programming assistants to solve their optimization models in Gurobipy and implement their simulation models with Scipy and Numpy. However, we do not require that students know how to code, let alone code in Python, before enrolling in the course. Of course, it is reasonable to expect that a self-selection effect might imply that there are many OMAI enrollees who are already strong coders (see Sec. 4), but it is certainly not the case that all OMAI students know how to code. (We have had students ask what a for loop is!) For those students with less coding experience, we believe that pair-programming is especially appropriate. Rather than receiving a chatbot's full implementation that is effectively a black box, obfuscating the model details at scale, the intimacy of line-by-line prompting essentially implies that students will build familiarity with individual functions and arguments as they work through a variety of examples. Thus, we choose to use only pair-programming AI assistance in OMAI to both keep modeling as the primary learning objective and help new coders learn relevant language details in a bite-sized fashion.

Both the model-first philosophy and the incorporation of pair-programming are reflected in the assessments and exercises in OMAI. Because practical implementation is a core value for a business education context, students are given several project-based assignments throughout the semester in which they must work through the full cycle of model formulation (in many cases, based on real-world data), implementation to code, and interpretation of the solution. On the assignments, students will be assessed on each of these fronts, and they have all resources available to them. On the exams (typically one for optimization and one for simulation), modeling and solving are decoupled and assessed separately through two different exam segments. One in-class segment asks students to develop a model for a specific scenario in pencil-and-paper fashion, without any resources. Then, a second at-home segment asks students to implement and solve a different specific model and then obtain insights from it. All resources, including pair-programming assistants, are available to students on the solving segment.

4. Insights From a Modeling Self-Assessment Survey of OM Students

In the four semesters that we have taught OMAI at our institution, we have observed anecdotally through learning evaluations and student interactions that the course has had impact: students have told us that the class has influenced their career paths and expanded their academic interests, and the growing enrollment suggests that the course has a positive reputation among students. We are encouraged by its success so far. (See Sec. 5 for our ideas on how we might improve the course in future semesters.) Nevertheless, this feedback is informal and not actually measured relative to its predecessor and parallel option, TOM. After all, students only take one course, and we have also received positive feedback from students who have taken TOM.

To measure the success of the course on its goals of expanding understanding, adoption, and advocacy of modeling, and moreover, to understand how the use of generative AI impacts these goals, we administered a survey to students in both TOM and OMAI at the end of the Spring 2025 semester. This study was reviewed by the University of Southern California Institutional Review Board and determined to be exempt from further review under Section: 46.104(d) (1) (Study ID UP-25-00209). The survey was entirely anonymous in both response and record of completion, students' participation was entirely voluntary, and, to prevent any potential conflict of interest among active instructors in that semester, only the first author was involved in constructing the survey, collecting the responses, and analyzing the data. Participants were recruited both in person during class time and via email to course rosters. A total of 250 respondents participated, with 156 from TOM and 94 from OMAI.

The survey contained four main sections of student self-assessment questions: three background questions on the students' experience prior to taking either TOM or OMAI, six questions on students' comfort and confidence for general modeling content in the overlap of the two courses, five questions on the effects of pair programming asked only to OMAI students, and two questions on students' comfort and confidence in the use of modeling concepts in a particular example. All of the survey questions were on a Likert-scale: on each question, respondents were presented with a statement and asked to select the option among "strongly disagree," "somewhat disagree," "neither agree nor disagree," "somewhat agree," and "strongly agree" that best described their agreement with the statement.

In the background section, respondents were asked to self-assess their experience with coding prior to taking TOM or OMAI, their prior experience with Python specifically, and their prior experience with pair-programming AI assistants such as GitHub Copilot. Then, the general course content section asked respondents to self-assess their comfort implementing LPs and IPs, their understanding of how LPs and IPs are used in decision-making, their comfort in using

LPs and IPs at work, their comfort incorporating uncertainty in business analysis and decision-making, their understanding of how mathematical models can be valuable for decision-making, and their confidence in developing a new model when faced with a new problem. In the pair-programming section of questions that was shown only to OMAI participants, respondents were asked to self-assess whether pair-programming assistants made it easier to implement and solve LPs and IPs, whether pair-programming assistants made it easier to incorporate uncertainty into decision-making, whether pair-programming helped them understand how they can use modeling to drive decision-making, whether pair-programming assistants expanded their comfort zone in implementing mathematical models, and whether pair-programming increased their confidence in solving new and complex business problems. Finally, the last section of the survey presented a specific problem scenario and asked respondents (from both courses) to self-assess their comfort in formulating an optimization model for that problem and their comfort in implementing and solving a model for that problem. The full statements of all 16 questions on the survey are available in Section A of the appendix, and Table 1 in the appendix contains the full distribution of response selections for each question in each of the two course populations.

Given that the survey focuses on modeling and emphasizes LPs, IPs, and Monte Carlo simulation, it should not be surprising for OMAI students to express higher comfort and confidence with this technical material. Indeed, this is what the survey results immediately reveal, and the contrast between TOM and OMAI can be striking. For instance, in the question on whether respondents report that they understand how LPs and IPs are used to make decisions and solve real-world problems (row Q5 of Table 1), 99% of OMAI respondents indicated agreement, with the majority showing strong agreement, and none indicated disagreement: 1 out of 94 respondents chose neither agree nor disagree (1.1%), 36 chose somewhat agree (38.3%), and 57 chose strongly agree (60.6%). By comparison, the TOM responses were dense across the spectrum of options: 28 out of 156 respondents chose strongly disagree (18.0%), 18 chose somewhat disagree (11.5%), 23 chose neither agree nor disagree (14.7%), 56 chose agree (35.9%), and 31 chose strongly agree (19.9%). The stark differences in response distributions from each course population are also apparent in the specific scenario questions. For instance, when asked about their comfort implementing and solving a given optimization model for the example problem (row Q16 of Table 1), TOM responses contained 7 strongly disagree (4.5%), 22 somewhat disagree (14.1%), 39 neither agree nor disagree (25.0%), 66 agree (42.3%), and 22 strongly agree (14.1%), whereas OMAI responses contained no strongly disagree (0%), 4 somewhat disagree (4.3%), 7 neither agree nor disagree (7.4%), 44 somewhat agree (46.8%), and 39 strongly agree (41.5%) selections.

Though there is the potential for Dunning-Kruger effects in self-reporting that could blur the distinction between the two groups, we believe these immediate results are not surprising for two

main reasons. First, OMAI is essentially entirely focused on these topics, and thus students have had ample practice and experience to build the comfort and confidence that these questions ask about – that is essentially the goal of the course, and thus what we hoped to see in the survey results. Second, as we have mentioned in the contrast of TOM and OMAI in Section 3, there is an inherent self-selection effect to this pair of courses: students with stronger prior technical background may be more inclined to enroll in the more technical course. By comparison to the first reason, the prior experience of OMAI students is not what we hope to be driving the successful outcomes of the course. With that in mind, let us take a closer look at the responses to the background questions in each course.

4.1. Stratifying Student Responses by Prior Coding Experience

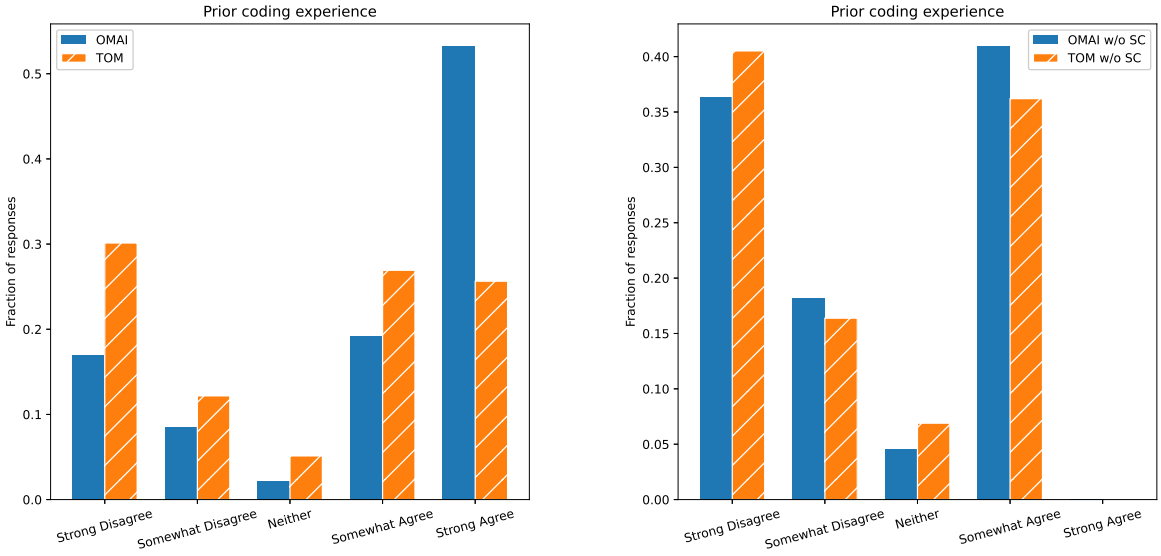

Figure 2 *Left*: Histogram of respondents' reported confidence with coding prior to taking the course. *Right*: Histogram of respondents' reported confidence with coding prior to taking the course, conditioned on not reporting strong confidence.

Recall that there were three background questions on the survey: prior coding experience, prior Python experience, and prior use of pair-programming AI assistants. In left-hand side of Figure 2, we plot histograms for the distribution of responses to the prior coding question in each of the courses (which is also available in row Q1 of Table 1). As one might expect, these distributions are quite different: notably, the OMAI population has a majority of respondents indicating that they had strong prior coding experience (53.2%). By comparison, barely a quarter of TOM respondents

reported strong prior coding experience (25.6%), and a plurality of TOM respondents selected strongly disagree for the question (30.1%).

However, one may also notice in the left-hand side of Figure 2 that the relative proportions of the bars for strongly disagree, somewhat disagree, neither, and somewhat agree are not so dissimilar across the TOM and OMAI distributions. This observation brings us to the right-hand side of Figure 2. This plot shows the histograms of the survey responses when removing those that indicated strong prior coding experience, meaning the conditional distributions given that the respondent's self-reported prior coding experience was less than strong. When controlling for prior coding experience in this way, a likeness of subpopulations emerges. In fact, the four conditional fractions of responses (from strongly disagree to somewhat agree) have an absolute difference of less than 0.05 in each selection ($[.364, .182, .045, .409]$ for OMAI and $[.405, .164, .069, .362]$ for TOM).

Interestingly, this split on strong coders finds that the course subpopulations also look more alike in the other background questions. For instance, the conditional distributions (from strongly disagree to strongly agree) of prior Python experience for non-strong-coders are $[.523, .091, .045, .227, 0.114]$ for OMAI and $[.517, .095, .043, .267, .078]$ for TOM, and, among these two groups of non-strong-coders, a very similar majority of respondents selected strongly disagree for prior pair-programming use (72.7% for OMAI and 70.7% for TOM). Then, similarities also appear among the strong-coder subpopulations. For instance, the prior Python distributions once again look quite similar, with $[.040, .020, .020, .140, .780]$ for OMAI and $[.077, .051, .000, .128, .744]$ for TOM, though, interestingly, a higher fraction of TOM strong coders indicate prior use of pair-programming assistants, with $[.380, .200, .020, .160, 0.240]$ for OMAI and $[.325, .100, .050, .125, 0.400]$ for TOM.

Following the recognition of these similar subpopulations of differing sizes within the OMAI and TOM groups, let us review the distributions of responses to the background questions conditioned on either being from strong coders or non-strong coders (i.e., somewhat agree or lower). The full conditional distributions for all questions can be found in Tables 2 and 3 in the appendix for without strong coders and for exclusively strong coders, respectively. As these tables show, both the more and less technical subpopulations appear to be better served in the new class, with OMAI respondents reporting higher comfort and confidence across the board. Let us highlight these findings in a few example questions.

First, let us return to the question regarding the use of LPs and IPs for decision-making which we summarized for the full population level at the start of this section (row Q5 in Tables 2 and 3). We now plot the distribution of responses in each subpopulation across each course in Figure 3, with non-strong coders on the left plot and strong coders on the right. Among non-strong coders in TOM, the responses are fairly uniform across the options ($[.233, .147, .147, .328, .147]$), whereas among

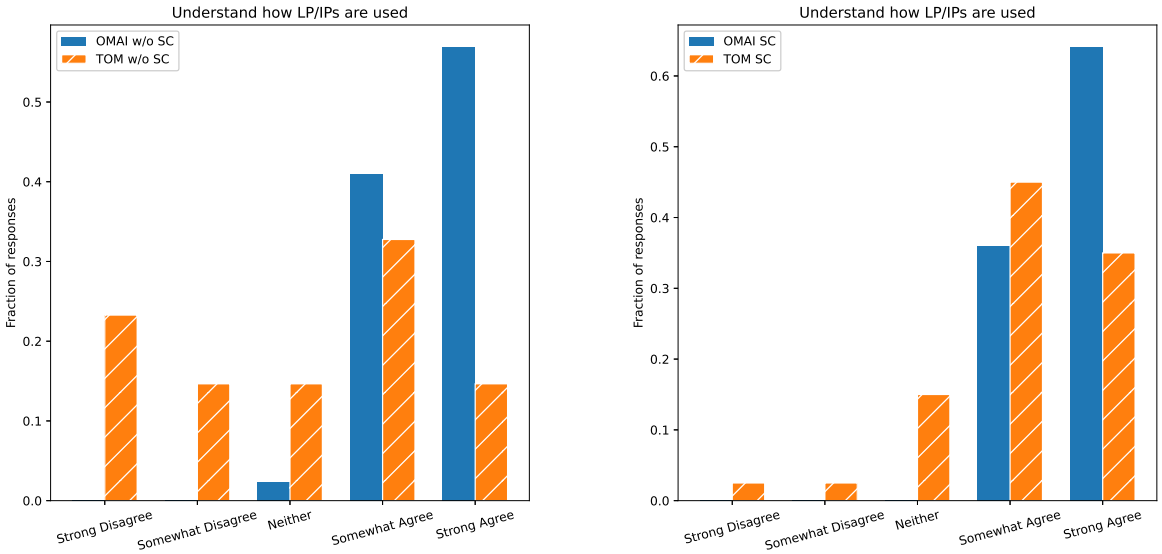

Figure 3 *Left*: Histogram of reported understanding use of LPs and IPs among non-strong coders. *Right*: Histogram of reported understanding use of LPs and IPs among strong coders.

non-strong coders in OMAI, a majority of respondents report strong confidence in understanding how LPs and IPs are used to solve real-world problems, with no respondents disagreeing with the statement ($[.000, .000, .023, .409, .568]$). For the strong coders, a majority of respondents in both courses report agreement, but in TOM the mode of selections is somewhat agree, with 20% of respondents not selecting either agree option. By comparison, all strong coders in OMAI agreed with the LP and IP usage statement, and 64% did so strongly.

These conditional distributions can also reveal dimensions on which the less technically experienced students particularly benefit. For example, consider the question on general appreciation of modeling as a decision-making framework (row Q8 in Tables 2 and 3), which we plot for each subpopulation in Figure 4. As one may notice on the right, the distribution of responses for strong coders is very similar across the two courses, suggesting that perhaps students with high levels of prior coding experience already appreciated the value of modeling before taking either course. However, on the left, we see that 82% of non-strong coders in OMAI reported strong agreement with the statement that they "understand how mathematical models can be valuable for decision making in real-world problems." Moreover, all OMAI respondents reported at least some form of agreement. By comparison, strong agreement has a plurality but not quite a majority among non-strong coders in TOM (47.4%), with 18% of respondents not selecting either agreement option. Thus, these survey results suggest that the new course may be more effective at conveying the value of mathematical modeling to students who had not been particularly familiar with models before the class.

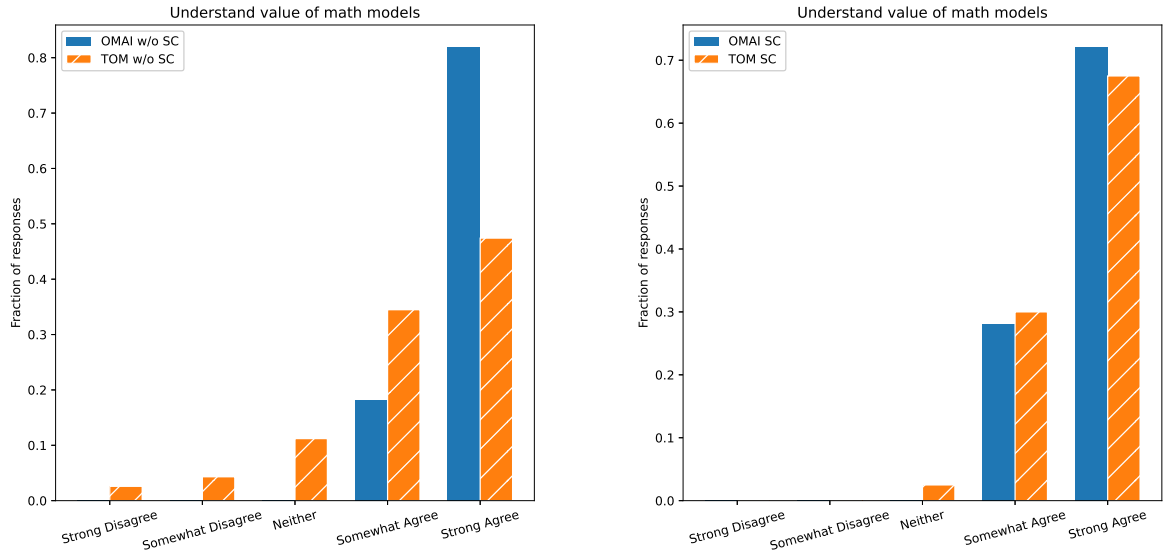

Figure 4 *Left*: Histogram of reported understanding of the value of mathematical models among non-strong coders.

Right: Histogram of reported understanding of the value of mathematical models among strong coders.

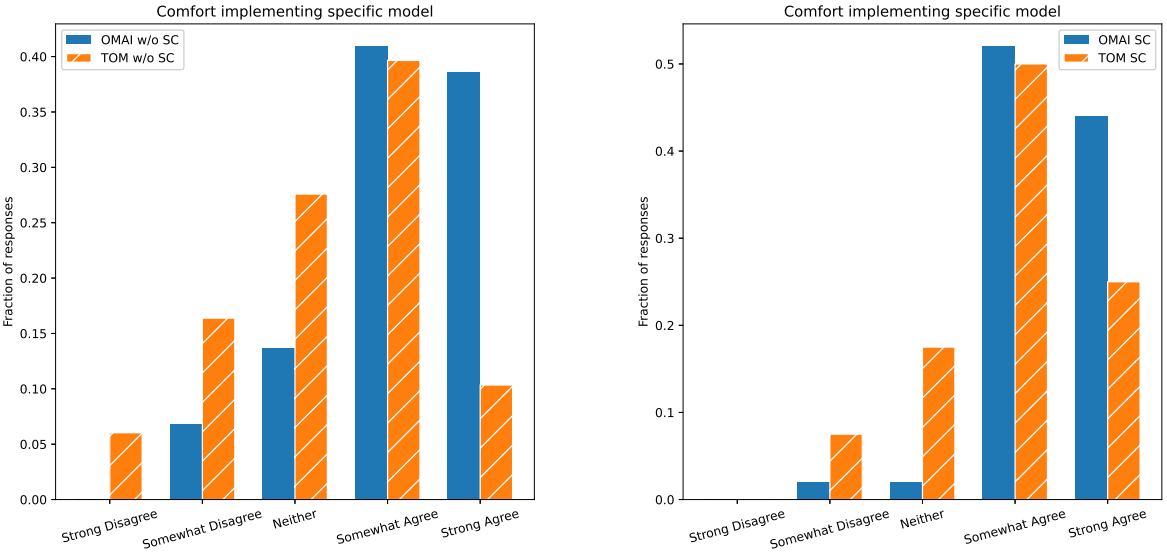

Figure 5 *Upper*: Histogram of reported comfort implementing and solving a given model for a specific example scenario among non-strong coders.

Lower: Histogram of reported comfort implementing and solving a given model for a specific example scenario among strong coders.

As a final example of the differences in conditional response distributions, let us return to the question on student's comfort implementing and solving a given optimization model for a specific example problem (row Q16 in Tables 2 and 3). These distributions are shown now in Figure 5. On the left, we see that 80% of non-strong coders in OMAI indicate some level of agreement that they could implement and solve a given model, with 39% indicating strong confidence in their ability to implement and solve. Though a similar fraction of TOM non-strong coders somewhat agree with the statement, only 10% feel strongly that they could, and 41% did not select either agreement option. Among strong coders, there is a similar pattern of nearly equivalent amounts of respondents somewhat agreeing across the two courses (52% for OMAI and 50% for TOM), with more OMAI strong coders strongly agreeing (44% versus 25%) and far fewer not selecting an agreement option at all (4% versus 25%). Hence, these survey results suggest that the new course is more effective in the delivery of OM modeling skills for students at both the less or the more technically experienced levels.

4.2. OMAI Students Report Increased Confidence from AI Pair-Programming

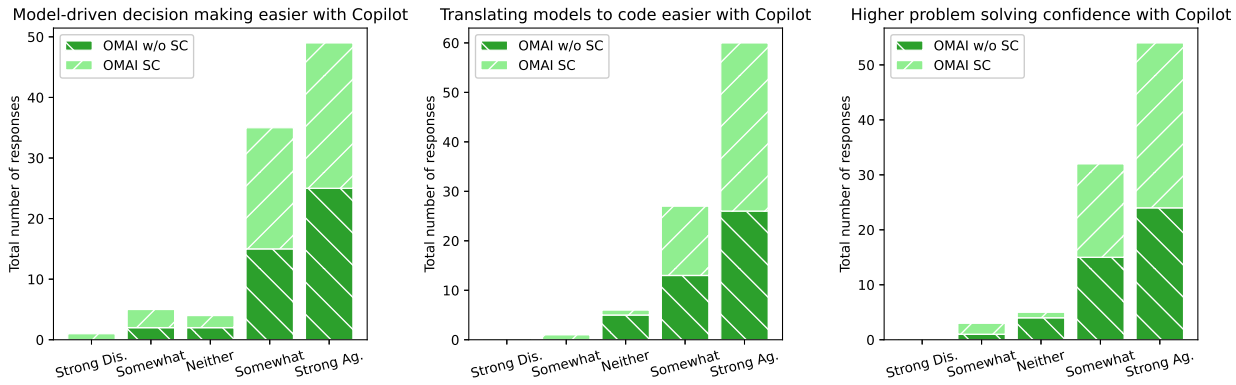

Figure 6 *Left*: Histogram of reported higher ease of model-driven decision making with Copilot. *Middle*: Histogram of reported higher ease in translating models to code with Copilot. *Right*: Histogram of reported higher problem solving confidence with Copilot.

As a final set of takeaways from these survey results, let us now consider the distributions of selections on the questions about the use of the pair-programming assistant GitHub Copilot in class, which were only asked to OMAI students (rows Q10 through Q14 in Tables 1, 2, and 3). Whether viewed at the full population or subpopulation levels, the majority of responses are positive on each of the five questions: students overall feel that pair-programming makes it easier to implement and solve LPs and IPs and easier to build simulations, pair-programming has helped them understand how to apply modeling to drive decision making and translate models into code,

and pair-programming has increased their confidence in solving new problems. We highlight three of these questions in Figure 6. In general, we interpret these survey results to mean that pair-programming is a beneficial use of AI for students learning modeling in undergraduate business education.

However, these responses do also suggest interesting sub-trends for the student subpopulations. In particular, agreement with the statements is not universal, and as can be seen through either Figure 6 or in Tables 1, 2, and 3, the strong coders consistently out-number the non-strong coders in disagreement to these questions. Though the sample size of this survey is too small to draw any strong conclusions away from this, these observations may at least suggest some hypotheses. For instance, it is well known that generative AI is prone to errors and hallucinations, and so it is not a guarantee that the code produced by a pair-programming assistant will be correct. In general, we feel that Incremental Prompting may be better equipped to manage this risk of error by comparison to wholesale code produced through one-shot chatbot prompting, but some risk exists regardless. Hence, some strong coders may be recognizing the potential flaws of over-reliance on generative AI here, and some may believe they are better off without it. Furthermore, as aligned with the findings in Bastani et al. (2025), students who have strong prior technical experience may also be recognizing that they could become overly dependent on the assistance generative AI and not fully learning how to implement and solve models on their own. That hypothesis is of particular concern to us, and the balance of class time, exercises, and assessments spent with and without AI is a key focus of our reflections from teaching this course so far.

5. Further Course Details, Best Practices, and Future Directions

In this final section of the paper, let us close with best practices, caveats, and future directions for using AI in undergraduate business classrooms to help with developing large-scale optimization and simulation models. To best facilitate adoption by other instructors, we organize these reflections as a series of topics.

Continual Emphasis on Critical Thinking: When using AI technology like GitHub Copilot in a classroom, a common phenomenon that we observed is that students sometimes abandon critical thinking and blindly accept the solution/code produced by Copilot, without checking whether the answer is correct. During the class, we continually emphasize to our students the importance of critical thinking and of continually assessing the code produced by AI technology. We repeatedly tell our students that they should *never* merely accept the solution offered by AI technology without scrutinizing it first. While the code produced by Copilot is correct most of the time; in many cases, the code is too complicated and lacks interpretability, and a simpler solution exists. As part of the Incremental Prompting framework, we always encourage our students to critique and assess the

quality of the solution produced by AI. In addition to continual emphasis on critical thinking, we found three helpful tactics, which are discussed in the next three topics below.

Focus on Model Development and Understanding Before Using any AI Technology: We structured OMAI so that, in every class session, we spent the first 60 minutes (in a 110-minute class) discussing the optimization/simulation models. During these 60 minutes, we do not use any AI technology. The time is focused exclusively on lecture, working on the whiteboard or on paper, and interactive discussion with students. We would discuss the underlying business problem with the students and engage with them via the Socratic method to identify relevant decision variables and constraints. Then, we would write out the entire optimization formulation on the whiteboard. Only after the students thoroughly understand the underlying optimization/simulation model will we then start implementing it in Python.

Continual Student Engagement While Using AI: When we implement optimization/simulation models in Python with GitHub Copilot, students are expected to work alongside us on their laptops, and we make sure to engage with students continually. After writing out each prompt and observing the code produced by Copilot, we would pause and engage with the students. Typical questions that we would ask students are "does this code make sense?", "how does this code match with the model we wrote up in the white board earlier?", and "how can we improve the code produced by Copilot?" If the code produced by Copilot is incorrect or a more elegant solution exists, we would use this as a teaching opportunity and show the students how to change the prompt so that Copilot would provide a different solution. Through this Incremental Prompting process, we encourage our students to think critically about the answer produced by Copilot, and we are confident that this process will ultimately enhance their understanding of the underlying model.

Teaching Students About Prompt Engineering and Error Correction: Providing precise and detailed prompts (aka prompt engineering) is extremely important. So, for each lesson, we spent a lot of time teaching students how to provide good prompts for Copilot and gave examples of good versus bad prompts (e.g., Figure 1). In our experience, Copilot works best with line-by-line prompting. So, rather than teaching students to ask the Copilot to "implement" the model, we teach them to provide Incremental Prompts. In particular, after we have developed the model on the whiteboard, we encourage the students to first write out the description of all the decision variables in plain but mathematically precise language, and prompt the Copilot to define these decision variables in Python with Gurobi.

After all the decision variables have been defined, we would then ask students to write out each constraint in plain language but with mathematically precise terms, and then prompt Copilot to translate the constraint into Python. Of course, we always check to make sure that the resulting constraint is consistent with the mathematical model that we have developed. During this process,

we also encouraged students to interact with Copilot through its chat feature, which allows students to ask more open-ended coding questions.

We want to emphasize that the process of building a good optimization or simulation model with Copilot is more than a single query or prompt. Instead, it is an iterative process, where the students need to know how to precisely describe their model and to understand the code suggested by Copilot, even though they do not need to remember syntax details of Python. When combined with instruction and guidance in the classroom, the use of Copilot enhances the students' understanding of the optimization/simulation models because they need to describe the models in detail and with precision.

Separation of Learning Objectives and Incorporation of AI Assistance: As we have discussed, a core component of the Incremental Prompting approach in OMAI is that modeling is an action of the student, and the output of their modeling becomes the granular input that they give to the pair-programming assistant. In this way, the learning objective of the course, modeling, is separated from what AI is used to do, coding and implementing. Because the nascent literature on AI usage has already documented skill regression in settings as diverse as high school students learning math (Bastani et al. 2025) and expert endoscopists conducting colonoscopies Budzyń et al. (2025), we feel that this separation is critical. The goal of the course is to teach students to model, not to teach students to code, thus we keep the AI usage focused on coding.

Teaching Math Concepts in Degree Programs That Are Not Math-Oriented: Even though the survey results discussed in Section 4 suggest that the modeling-focused class has been well-received across the population of students, we believe that it is important to remember that business students may not select their major consciously seeking math, and thus there are many students who greet these topics with some degree of math anxiety, perhaps more so than in engineering or other openly STEM degree programs. The education literature has shown that is important in such settings to encourage students to maintain a growth mindset (e.g., Boaler et al. 2021, Bui et al. 2023). Thus, we believe it is particularly important to structure the course in an interactive manner. This naturally includes the Incremental Prompting approach to AI, but it should also underlie the basic approach to modeling in class activities. Optimization and simulation are certainly rich topics mathematically, but their roots are in broadly accessible concepts. For example, in optimization, the linear functions that make up the LP's and IP's covered in OMAI are not only the friendliest function class for solvers, but also, in many aspects, the friendliest for new modelers. By consistently scaffolding from such familiar starting points throughout the semester, one can design the instruction of advanced use and combination of these elementary concepts into complex models to be both inviting and enticing across backgrounds.

Determining Daily Flow of Class Sessions and Fraction of Time Spent on Coding: Typically, the class flow for exercises and cases in the OMAI courses has been that, roughly speaking, we first we all model together as a class, and then we all implement together as a class. This has worked well overall, but as evidenced from the survey results in Section 4, there is a wide variety of student backgrounds, and different degrees of technical experience may be better suited to different amounts of in-class implementation time. For instance, if the students already all have a strong coding background, then instructors can perhaps focus nearly exclusively on modeling in class and leave implementation as an at-home task that is then recapped at the start of the next day. For this group, too much focus on pair-programming assistance could actually be counterproductive. By comparison, if only a few students have a strong prior coding background, then instructors might instead devote significant class time to teaching Copilot and the ideas of Incremental Prompting; this is largely how we have approached the course so far. Then, if population has a variety of backgrounds (which our survey results suggest may be most likely), then it may actually work best to have some Copilot demonstrations in class and some others in attendance-optional formats like recitations or in asynchronous material like flipped-classroom videos. Indeed, after reviewing the survey results from our own students, this is a format with which we intend to experiment in the coming semesters.

Teaching Model Validation As an Essential Skill: As a final reflection, let us note that perhaps our foremost focus for improvement in current and future iterations of the course is to further develop the course's content on model validation. On a fundamental level, we feel that knowing how to audit a model for actually representing reality as the modeler intended is a critical skill for mature modelers and an essential one for true practitioners. Given that our students are also drawing upon AI to implement their models, even if they do so incrementally, it only becomes more important that they can evaluate whether the end result functions as desired. Model validation exists in the class to some degree through topics like scrutinizing pair-programming output, sensitivity analysis, robustness checks, and reductions to edge cases or special settings, but it is the educational aspect we are most focused on developing further as the course matures. To that end, we welcome any feedback or suggestions from the community on how to do this well, and we are generally eager to collaborate and exchange ideas for ways to better teach modeling in undergraduate business education in the future.

Acknowledgements

We are grateful for the generous support of this study and its associated educational efforts by the National Science Foundation Division of Civil, Mechanical, and Manufacturing Innovation (Award #CMMI-2441387, received by A. Daw). We are also grateful to colleagues who have shared

materials with us from which we have based the new course, including Professor Amy Ward, who shared simulation materials, and Professor Huseyin Topaloglu, who shared optimization materials. Following that spirit, we would be very happy to share any amount of materials by request from any prospective future instructors.[1]

References

AhmadiTeshnizi A, Gao W, Brunborg H, Talaei S, Lawless C, Udell M (2025) Optimus-0.3: Using large language models to model and solve optimization problems at scale URL https://arxiv.org/abs/2407.19633.

Bachiri YA, Mouncif H, Bouikhalene B (2023) Artificial intelligence empowers gamification: Optimizing student engagement and learning outcomes in e-learning and moocs. *International Journal of Engineering Pedagogy* 13(8).

Bastani H, Bastani O, Sungu A, Ge H, Kabakcı Ö, Mariman R (2025) Generative AI without guardrails can harm learning: Evidence from high school mathematics. *Proceedings of the National Academy of Sciences* 122(26):e2422633122.

Belkina M, Daniel S, Nikolic S, Haque R, Lyden S, Neal P, Grundy S, Hassan GM (2025) Implementing generative AI (GenAI) in higher education: A systematic review of case studies. *Computers and Education: Artificial Intelligence* 100407.

Bertsimas D, Freund R (2004) *Data, Models, and Decisions: The Fundamentals of Management Science* (Dynamic Ideas), ISBN 9780975914601, URL https://books.google.com/books?id=NPKPPgAACAAJ.

Bien J, Burgos M, Cardon P, Carnevale P, Chen F, Guo Y, Gupta V, Henneman T, Hill M, Ho G, et al. (2024) Embracing AI in business education: A case study of USC Marshall School of Business .

Bien J, Mukherjee G (2025) Generative AI for Data Science 101: Coding without learning to code. *Journal of Statistics and Data Science Education* 33(2):129–142.

Boaler J, Dieckmann JA, LaMar T, Leshin M, Selbach-Allen M, Pérez-Núñez G (2021) The transformative impact of a mathematical mindset experience taught at scale. *Frontiers in Education*, volume 6 (Frontiers Media SA).

Bray RL (2025) A tutorial on teaching data analytics with generative ai. *INFORMS Journal on Applied Analytics* .

Budzyń K, Romańczyk M, Kitala D, Kołodziej P, Bugajski M, Adami HO, Blom J, Buszkiewicz M, Halvorsen N, Hassan C, et al. (2025) Endoscopist deskilling risk after exposure to artificial intelligence in colonoscopy: a multicentre, observational study. *The Lancet Gastroenterology & Hepatology* .

[1] A sample syllabus can be found at https://faculty.marshall.usc.edu/Andrew-Daw/BUAD313_Syllabus-fall23.pdf

Bui P, Pongsakdi N, McMullen J, Lehtinen E, Hannula-Sormunen MM (2023) A systematic review of mindset interventions in mathematics classrooms: What works and what does not? *Educational Research Review* 100554.

Bynum ML, Hackebeil G, Hart WE, Laird CD, Nicholson BL, Siirola JD, Watson JP, Woodruff DL (2020) Pyomo-optimization modeling in python 3rd ed. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

Cachon G, Terwiesch C (2008) *Matching supply with demand*, volume 20012 (McGraw-Hill Publishing New York).

Cai J, Mandelbaum A, Nagaraja CH, Shen H, Zhao L (2019) Statistical Theory Powering Data Science. *Statistical Science* 34(4):669 – 691, URL http://dx.doi.org/10.1214/19-STS754.

Chang CY, Kuo SY, Hwang GH (2022) Chatbot-facilitated nursing education. *Educational Technology & Society* 25(1):15–27.

Chemers MM, tze Hu L, Garcia BF (2001) Academic self-efficacy and first year college student performance and adjustment. *Journal of Educational Psychology* 93:55–64, URL https://api.semanticscholar.org/CorpusID:55236689.

Chen F, Samroengraja R (2000) The stationary beer game. *Production and Operations Management* 9(1):19–30.

Chen Z, Xiong P (2023) RSOME in Python: An open-source package for robust stochastic optimization made easy. *INFORMS Journal on Computing* 35(4):717–724.

Corbett CJ, Van Wassenhove LN (1993) The natural drift: What happened to operations research? *Operations Research* 41(4):625–640.

Deslauriers L, McCarty LS, Miller K, Callaghan K, Kestin G (2019) Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom. *Proceedings of the National Academy of Sciences* 116(39):19251–19257, URL http://dx.doi.org/10.1073/pnas.1821936116.

Dobson G, Shumsky R (2006) Web-based simulations for teaching queueing, Little's Law, and inventory management. *INFORMS Transactions on Education* 7(1):106–124.

Dong L, Tang X, Wang X (2025) Examining the effect of artificial intelligence in relation to students' academic achievement in classroom: A meta-analysis. *Computers and Education: Artificial Intelligence* 100400.

Duckworth AL, Yeager DS (2015) Measurement matters: Assessing personal qualities other than cognitive ability for educational purposes. *Educational researcher* 44(4):237–251.

Dunning I, Gupta V, King A, Kung J, Lubin M, Silberholz J (2015) A course on advanced software tools for operations research and analytics. *INFORMS Transactions on Education* 15(2):169–179.

Dunning I, Huchette J, Lubin M (2017) Jump: A modeling language for mathematical optimization. *SIAM review* 59(2):295–320.

Eckstein J, Riedmueller ST (2002) YASAI: Yet another add-in for teaching elementary Monte Carlo simulation in Excel. *INFORMS Transactions on Education* 2(2):12–26.

Evans JR (2000) Spreadsheets as a tool for teaching simulation. *Informs transactions on education* 1(1):27–37.

Falchikov N, Boud D (1989) Student self-assessment in higher education: A meta-analysis. *Review of Educational Research* 59(4):395–430, ISSN 00346543, 19351046, URL http://www.jstor.org/stable/1170205.

Fourer R, Gay DM, Kernighan BW (1990) AMPL: A mathematical programming language. *Management Science* 36(5):519–554.

Gorman MF (2021) INFORMS Journal on Applied Analytics editor's statement: The critical role of applied research in analytics. *INFORMS Journal on Applied Analytics* 51(1):1–5.

Grant M, Boyd S, Ye Y (2006) Disciplined convex programming. *Global optimization: From theory to implementation*, 155–210 (Springer).

Griffin P (2007) The use of classroom games in management science and operations research. *INFORMS Transactions on Education* 8(1):1–2.

Hill RR (2002) Process simulation in excel for a quantitative management course. *INFORMS Transactions on Education* 2(3):75–84.

Irwin V, Wang K, Tezil T, Zhang J, Filbey A, Jung J, Mann FB, Dilig R, Parker S (2023) Report on the Condition of Education 2023. *National Center for Education Statistics* .

Jensen PA, Bard JF (2002) *Operations research: Models and methods* (John Wiley & Sons).

Kan AHR (1989) The future of operations research is bright. *European journal of operational research* 38(3):282–285.

Kruger J, Dunning D (1999) Unskilled and unaware of it: how difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of personality and social psychology* 77(6):1121.

Lawless C, Schoeffer J, Le L, Rowan K, Sen S, St Hill C, Suh J, Sarrafzadeh B (2024) "I want it that way"': Enabling interactive decision support using large language models and constraint programming. *ACM Transactions on Interactive Intelligent Systems* 14(3):1–33.

Liberatore MJ, Luo W (2010) The analytics movement: Implications for operations research. *Interfaces* 40(4):313–324.

Lofberg J (2004) YALMIP: A toolbox for modeling and optimization in MATLAB. *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, 284–289 (IEEE).

Martin K (2010) Tutorial: COIN-OR: Software for the or community. *Interfaces* 40(6):465–476.

Mason AJ (2012) OpenSolver-an open source add-in to solve linear and integer progammes in Excel. *Operations Research Proceedings 2011: Selected Papers of the International Conference on Operations Research (OR 2011), August 30-September 2, 2011, Zurich, Switzerland*, 401–406 (Springer).

Mason AJ (2013) SolverStudio: A new tool for better optimisation and simulation modelling in Excel. *INFORMS Transactions on Education* 14(1):45–52.

Mortenson MJ, Doherty NF, Robinson S (2015) Operational research from Taylorism to terabytes: A research agenda for the analytics age. *European Journal of Operational Research* 241(3):583–595.

Multon KD, Brown SD, Lent RW (1991) Relation of self-efficacy beliefs to academic outcomes: A meta-analytic investigation. *Journal of counseling psychology* 38(1):30.

Pachamanova D (2006) Introducing integer modeling with Excel solver. *INFORMS Transactions on Education* 7(1):88–98.

Pasin F, Giroux H (2011) The impact of a simulation game on operations management education. *Computers & Education* 57(1):1240–1254.

Ramamonjison R, Yu TT, Li R, Li H, Carenini G, Ghaddar B, He S, Mostajabdaveh M, Banitalebi-Dehkordi A, Zhou Z, Zhang Y (2023) Nl4opt competition: Formulating optimization problems based on their natural language descriptions URL https://arxiv.org/abs/2303.08233.

Rardin RL (1998) *Optimization in operations research*, volume 166 (Prentice Hall Upper Saddle River, NJ).

Sitzmann T, Ely K, Brown KG, Bauer KN (2010) Self-assessment of knowledge: A cognitive learning or affective measure? *Academy of Management Learning & Education* 9(2):169–191.

Snider B, Balakrishnan J (2013) Lessons learned from implementing web-based simulations to teach operations management concepts. *INFORMS Transactions on Education* 13(3):152–161.

Sodhi MS, Tang CS (2008) The OR/MS ecosystem: Strengths, weaknesses, opportunities, and threats. *Operations Research* 56(2):267–277.

Wang L, Zhao M (2024) Can artificial intelligence technology promote the improvement of student learning outcomes?——meta analysis based on 50 experimental and quasi experimental studies (EAI), URL http://dx.doi.org/10.4108/eai.29-3-2024.2347685.

Xiao Z, Zhang D, Wu Y, Xu L, Wang YJ, Han X, Fu X, Zhong T, Zeng J, Song M, Chen G (2024) Chain-of-experts: When LLMs meet complex operations research problems. *The Twelfth International Conference on Learning Representations*, URL https://openreview.net/forum?id=HobyL1B9CZ.

Yan L, Sha L, Zhao L, Li Y, Martinez-Maldonado R, Chen G, Li X, Jin Y, Gašević D (2024) Practical and ethical challenges of large language models in education: A systematic scoping review. *British Journal of Educational Technology* 55(1):90–112, URL http://dx.doi.org/https://doi.org/10.1111/bjet.13370.

Zheng L, Niu J, Zhong L, Gyasi JF (2023) The effectiveness of artificial intelligence on learning achievement and learning perception: A meta-analysis. *Interactive Learning Environments* 31(9):5650–5664, URL http://dx.doi.org/10.1080/10494820.2021.2015693.

Appendix

A. Survey Questions

A.1. Background and Prior Technical Experience

All students were asked to rate their agreement the following statements regarding their background prior to taking the OM course:

Q1: "I had prior experience with coding before taking BUAD 311 or BUAD 313."

Q2: "I had prior experience with coding specifically in Python before taking BUAD 311 or BUAD 313."

Q3: "I had prior experience with using pair-programming generative AI assistants (i.e., GitHub Copilot) for coding before taking BUAD 311 or BUAD 313."

A.2. Self-Assessment of General Course Content

Additionally, all students were asked to rate their agreement with the following statements regarding their general comfort the mathematical modeling concepts common in both course:

Q4: "I feel comfortable implementing optimization models such as linear programs (LPs) and integer programs (IPs)."

Q5: "I understand how optimization models such as linear programs (LPs) and integer programs (IPs) are used to make decisions and solve real-world problems."

Q6: "I would feel comfortable using optimization models such as linear programs (LPs) and integer programs (IPs) in projects during an internship or full-time employment, either individually or as part of a team."

Q7: "I feel comfortable incorporating uncertainty in my analysis of business problems and evaluating the impact of the uncertainty on my decisions."

Q8: "I understand how mathematical models can be valuable for decision making in real-world problems."

Q9: "When faced with a business problem that does not fit with a standard formula, I feel confident that I can think through the problem, develop a mathematical model, and derive an effective decision for the underlying problem."

A.3. Self-Assessment of Pair Programming Usage

Students in the advanced course (BUAD 313) were asked to rate their agreement with the following statements regarding their experience with pair-programming generative AI assistants (i.e., GitHub Copilot) in the context of the course:

Q10: "Pair-programming generative AI assistants (i.e., GitHub Copilot) make it easier for me to implement and solve optimization models such as linear programs (LPs) and integer programs (IPs)."

Q11: "Pair-programming generative AI assistants (i.e., GitHub Copilot) make it easier for me to incorporate uncertainty into decision making through simulation."

Q12: "Pair-programming generative AI assistants (i.e., GitHub Copilot) have helped me understand how I can apply mathematical modeling to drive decision-making for real-world problems."

Q13: "Pair-programming generative AI assistants (i.e., GitHub Copilot) have expanded my comfort zone for translating mathematical models into scientific computing languages and libraries."

Q14: "Pair-programming generative AI assistants (i.e., GitHub Copilot) have increased my confidence in solving complex business problems that I have never seen before."

A.4. Self-Assessment of Specific Course Content and Usage

Finally, all students were asked to rate their agreement with two statements regarding their comfort with developing a model and implementing it for the following specific scenario regarding school transportation planning:

"The Santa Barbara Unified School District (SBUSD) manages 4 elementary schools collectively serving 12 distinct neighborhoods. Each elementary school offers 6 grade levels, from kindergarten to fifth grade.

In anticipation of the next academic year, SBUSD has compiled the following information:

- The number of students each elementary school can support at each grade level,
- The actual number of students at each grade level from each neighborhood,
- The distance from each neighborhood to each school."

The survey then presented the two following questions about this scenario:

Q15: "Given exact numbers for the information described above and ample time to think, I feel comfortable formulating an optimization model (i.e., identifying decision variables, objective function, and constraints) to assign elementary-age students in Santa Barbara to schools in order to minimize the total distance traveled by students to school each day."

Q16: "Given an optimization problem formulation that is based on the information described above, I feel comfortable implementing the model and solving for the school assignment which minimizes SBUSD student travel."

Table 1 Summary statistics of selected survey responses for the two course populations.

	OMAI					TOM						
	S.D.	D.	N.	A.	S.A.	S.D.	D.	N.	A.	S.A.	χ^2	p-value
Q1	0.170	0.085	0.021	0.191	0.532	0.301	0.122	0.051	0.269	0.256	19.894	0.001
Q2	0.266	0.053	0.032	0.181	0.468	0.406	0.084	0.032	0.232	0.245	13.587	0.009
Q3	0.543	0.149	0.043	0.128	0.138	0.609	0.147	0.077	0.064	0.103	4.865	0.301
Q4	0.011	0.064	0.117	0.585	0.223	0.244	0.212	0.199	0.263	0.083	55.266	0.000
Q5	0.000	0.000	0.011	0.383	0.606	0.179	0.115	0.147	0.359	0.199	66.937	0.000
Q6	0.000	0.117	0.170	0.489	0.223	0.218	0.205	0.192	0.276	0.109	35.869	0.000
Q7	0.011	0.128	0.191	0.468	0.202	0.083	0.173	0.256	0.359	0.128	11.177	0.025
Q8	0.000	0.000	0.000	0.234	0.766	0.019	0.032	0.090	0.333	0.526	20.709	0.000
Q9	0.011	0.096	0.181	0.436	0.277	0.071	0.269	0.167	0.346	0.147	19.347	0.001
Q10	0.000	0.000	0.032	0.202	0.766	-	-	-	-	-	-	-
Q11	0.000	0.021	0.053	0.362	0.564	-	-	-	-	-	-	-
Q12	0.011	0.053	0.043	0.372	0.521	-	-	-	-	-	-	-
Q13	0.000	0.011	0.064	0.287	0.638	-	-	-	-	-	-	-
Q14	0.000	0.032	0.053	0.340	0.574	-	-	-	-	-	-	-
Q15	0.000	0.043	0.032	0.521	0.404	0.045	0.097	0.200	0.516	0.142	35.319	0.000
Q16	0.000	0.043	0.074	0.468	0.415	0.045	0.141	0.250	0.423	0.141	37.810	0.000

B. Tables of Survey Results

B.1. Full Course Populations

Table 1 provides the distributions of response selections from all respondents within both course populations across all 16 questions. For each question that was asked to all respondents (Q1 through Q9, Q15, and Q16), the statistic and p-value for a chi-squared test across the two course distribution are reported. Please note Q2 and Q15 each received 155 responses from TOM, whereas all other questions were answered by all 156 respondents from TOM. All 94 respondents from OMAI answered all 16 questions.

B.2. Stratified by Prior Coding Experience

Then, Table 2 provides the distributions of response selections from the non-strong coder respondents within both course populations across all 16 questions. That is, the distributions shown here are calculated by removing responses that selected "Strong Agree" for Q1. For each question that was asked to all respondents (Q1 through Q9, Q15, and Q16), the statistic and p-value for a chi-squared test across the two course distribution are reported. All questions received 44 responses from the OMAI respondents without strong coders and 116 responses from the TOM respondents without strong coders, except for Q2, which received 115 responses from TOM without strong coders.

Finally, Table 3 provides the distributions of response selections from the exclusively strong coder respondents within both course populations across all 16 questions. That is, the distributions shown here are calculated by including only the responses that selected "Strong Agree" for Q1.

Table 2 Summary statistics of selected survey responses for the two course populations without respondents with a strong prior coding background.

	OMAI w/o Strong Coders					**TOM w/o Strong Coders**						
	S.D.	D.	N.	A.	S.A.	S.D.	D.	N.	A.	S.A.	χ^2	p-value
Q1	0.364	0.182	0.045	0.409	-	0.405	0.164	0.069	0.362	-	0.671	0.880
Q2	0.523	0.091	0.045	0.227	0.114	0.517	0.095	0.043	0.267	0.078	0.684	0.953
Q3	0.727	0.091	0.068	0.091	0.023	0.707	0.164	0.086	0.043	0.000	5.257	0.262
Q4	0.023	0.045	0.227	0.545	0.159	0.302	0.224	0.198	0.233	0.043	32.493	0.000
Q5	0.000	0.000	0.023	0.409	0.568	0.233	0.147	0.147	0.328	0.147	43.246	0.000
Q6	0.000	0.045	0.295	0.545	0.114	0.284	0.224	0.181	0.241	0.069	30.161	0.000
Q7	0.023	0.114	0.227	0.477	0.159	0.112	0.190	0.267	0.362	0.069	8.040	0.090
Q8	0.000	0.000	0.000	0.182	0.818	0.026	0.043	0.112	0.345	0.474	17.430	0.002
Q9	0.000	0.136	0.182	0.432	0.250	0.086	0.293	0.198	0.328	0.095	13.532	0.009
Q10	0.000	0.000	0.045	0.250	0.705	-	-	-	-	-	-	-
Q11	0.000	0.023	0.068	0.341	0.568	-	-	-	-	-	-	-
Q12	0.000	0.045	0.045	0.341	0.568	-	-	-	-	-	-	-
Q13	0.000	0.000	0.114	0.295	0.591	-	-	-	-	-	-	-
Q14	0.000	0.023	0.091	0.341	0.545	-	-	-	-	-	-	-
Q15	0.000	0.068	0.068	0.477	0.386	0.052	0.104	0.243	0.522	0.078	26.350	0.000
Q16	0.000	0.068	0.136	0.409	0.386	0.060	0.164	0.276	0.397	0.103	21.490	0.000

Table 3 Summary statistics of selected survey responses for the two course populations for only respondents with a strong prior coding background.

	OMAI Strong Coders					**TOM Strong Coders**						
	S.D.	D.	N.	A.	S.A.	S.D.	D.	N.	A.	S.A.	χ^2	p-value
Q1	-	-	-	-	1.000	-	-	-	-	1.000	0.000	1.000
Q2	0.040	0.020	0.020	0.140	0.780	0.077	0.051	0.000	0.128	0.744	2.008	0.734
Q3	0.380	0.200	0.020	0.160	0.240	0.325	0.100	0.050	0.125	0.400	4.235	0.375
Q4	0.000	0.080	0.020	0.620	0.280	0.075	0.175	0.200	0.350	0.200	16.413	0.003
Q5	0.000	0.000	0.000	0.360	0.640	0.025	0.025	0.150	0.450	0.350	14.107	0.007
Q6	0.000	0.180	0.060	0.440	0.320	0.025	0.150	0.225	0.375	0.225	6.858	0.144
Q7	0.000	0.140	0.160	0.460	0.240	0.000	0.125	0.225	0.350	0.300	1.489	0.685
Q8	0.000	0.000	0.000	0.280	0.720	0.000	0.000	0.025	0.300	0.675	1.345	0.510
Q9	0.020	0.060	0.180	0.440	0.300	0.025	0.200	0.075	0.400	0.300	5.510	0.239
Q10	0.000	0.000	0.020	0.160	0.820	-	-	-	-	-	-	-
Q11	0.000	0.020	0.040	0.380	0.560	-	-	-	-	-	-	-
Q12	0.020	0.060	0.040	0.400	0.480	-	-	-	-	-	-	-
Q13	0.000	0.020	0.020	0.280	0.680	-	-	-	-	-	-	-
Q14	0.000	0.040	0.020	0.340	0.600	-	-	-	-	-	-	-
Q15	0.000	0.020	0.000	0.560	0.420	0.025	0.075	0.075	0.500	0.325	7.193	0.126
Q16	0.000	0.020	0.020	0.520	0.440	0.000	0.075	0.175	0.500	0.250	9.792	0.020

For each question that was asked to all respondents (Q1 through Q9, Q15, and Q16), the statistic and p-value for a chi-squared test across the two course distribution are reported. All questions received 50 responses from OMAI strong coders and 40 responses from TOM strong coders, except for Q2, which received 39 responses from TOM strong coders.