EFFICIENT HUMAN-MACHINE WORK TRANSFER THROUGH LATENT STRUCTURE DECOMPOSITION

by

Gaoping Huang

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Electrical and Computer Engineering

West Lafayette, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Alexander J. Quinn, Chair

School of Electrical and Computer Engineering

Dr. Karthik Ramani

School of Mechanical Engineering

Dr. Hong Z. Tan

School of Electrical and Computer Engineering

Dr. Samuel P. Midkiff

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

This dissertation is dedicated to my parents who nurse me with affections and love so that I can pursue a dream with courage.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor Dr. Quinn for his full support, understanding, and encouragement throughout my research. When I first started my research, I was novice to programming and interface design. Through his patient advisory, I learned to pay attention to good coding style and design details. Meanwhile, I have been deeply influenced by his consistent effort in maintaining research ethics and research integrity. Besides, his humor and good temper make me feel relaxed in all of our conversation, both on academic topics and daily lives.

I would also like to thank Dr. Ramani for his support, insight, and assistance that led to the completion of my research. I am often impressed by how fast he can reply to the emails, how exciting when he depicts the future work, and how efficiently he can coordinate different people to get things done.

I am furthermore appreciative to Dr. Tan, Dr. Midkiff, and Dr. Xiaozhu Lin for their advices and caring. I am always encouraged and inspired during our short conversations.

My thanks also go to my colleagues in Human-Powered System Lab—Meng-Han Wu, VK Chaithanya Manam, and Abdullah Alshaibani. Thanks to you, I was able to experience cultural differences in a pleasant way and study without being lonely.

Moreover, I would like to thank all the folks in C-Design Lab. You guys make me understand how to work like a team. In particular, I would like to thank Xun Qian, Tianyi Wang, and Zhengzhe Zhu. My life becomes more interesting when we play games and eat hotpot together.

I would also thank my roommates–Jiexin Duan and Wei Deng–who are considerate of others so that the work-from-home life during COVID-19 is pleasant and in harmony.

I was supported by Frederick N. Andrews Fellowship from Purdue University and research assistant funding from Dr. Quinn. These projects were in part supported by NSF under the grants FW-HTF 1839971, OIA 1937036, and CRI 1729486. You all make graduate school possible - thanks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AMT	Amazon Mechanical Turk
AR	Augmented Reality
IoT	Internet of Things
RDF	Resource Description Framework
ROS	Robot Operating System

ABSTRACT

When humans delegate tasks—whether to human workers or robots—they do so either to trade money for time, or to leverage additional knowledge and capabilities. For complex tasks, however, describing the work to be done requires substantial effort, which reduces the benefit to the *requester* who delegates tasks. On one hand, human workers—e.g., crowd workers, friends or colleagues on social network, factory workers—have diverse knowledge and level of commitment, making it difficult to achieve joint efforts towards the requester's goal. In contrast, robots and machines have clearly defined capabilities and full commitment, but the requester lacks an efficient way to coordinate them for flexible workflows.

This dissertation presents a series of workflows and systems to enable efficient work transfer to human workers or robots. First, I present *BlueSky*, a system that can automatically coordinate hundreds of crowd workers to enumerate ideas for a given topic. The latent structure of the idea enumeration task is decomposed into a three-step workflow to guide the crowd workers. Second, I present *CoStory*, a system that requests alternative designs from friends or colleagues by decomposing the design task into hierarchical chunks. Third, I present *AdapTutAR*, a system that delegates machine operation tasks to workers through adaptive Augmented Reality tutorials. Finally, I present *Vipo*, a system that allows requesters to customize tasks for robots and smart machines through spatial-visual programming. This dissertation demonstrates that decomposing latent task structure enables task delegation in an on-demand, scalable, and distributed way.

1. INTRODUCTION

Delegating tasks is a common practice to save time or to leverage additional knowledge and capabilities. This dissertation focuses on two types of tasks that will benefit from task delegation. First, it focuses on delegating creative tasks to human workers. Creative tasks can take advantage of human workers' diversity and creativity. Second, this dissertation focuses on delegating routine but customizable tasks to factory workers, robots and machines. In manufacturing industry, robots and machines are widely used to improve precision and productivity, which may or may not need factory workers as operators.

While requesters may benefit from delegation, describing the work to be done requires substantial effort, which reduces the benefit. In general, to achieve successful task delegation, several key challenges need to be resolved, including division of tasks and management of dependencies [1], [2]. The task division challenge includes the consideration of how to divide subtasks in a meaningful and cohesive way so that the workers have enough context to work with. The managerial challenge includes the consideration of how to manage the interfaces between subtasks, how to match the subtasks with suitable workers, and what mechanisms are appropriate to achieve enough collaboration between workers. This latter challenge is closely related to communication [3], [4].

Special care should be given in resolving the above challenges when it comes to delegating creative tasks to human workers and delegating routine tasks to factory workers, robots and machines. On one hand, creative tasks are delegated to a large group of human workers, including crowd workers, friends, or colleagues. Those human workers have diverse level of knowledge, motivation and commitment to the creative tasks to be done. Besides, those workers behave in an on-demand, scalable, and distributed manner. Such characteristics lead to a unique set of design considerations, in terms of incentive design, quality control, task division, and task difficulty. In contrast, routine tasks are delegated to factory workers, robots and machines with clearly defined capabilities. Factory workers are expected to rapidly master new routines of operating machines to meet increasing demand of customizable and self-configuring production [5], [6]. This requirement poses challenges to train factory workers in an efficient and scalable manner, even when they have different

expertise. Meanwhile, robots and machines can achieve high precision robustly, but they typically behave in a fixed/predefined way. This limitation further requires factory workers to find an efficient approach to leveraging the autonomy of robots and machines to accomplish flexible/customizable tasks.

This dissertation develops delegation strategies to resolve the above challenges and enable efficient task delegation. First, I propose that identifying the latent structure of tasks can help decomposing complex tasks into simpler, self-contained subtasks. Second, I design workflows to better coordinate less-organized human workers. Third, I build a tutoring system that helps factory workers learn new machine operation tasks while adapting to their performance and prior knowledge. Finally, I introduce a spatial-visual programming to easily customize flexible workflows for robots and machines. These contributions are critical to the design of task delegation system: they shed lights on the task delegation from simple to complex, from creative to routine, and from crowd workers, friends, colleagues to factory workers, robots and machines.

The dissertation develops a series of prototypes, each of which demonstrates the aforementioned delegation strategies (fully or partially), as described below.

1.1 Human-centered and Machine-centered Task Delegation

In human-centered creative tasks and machine-centered customizable tasks, humans and machines have some commonality and difference.

Both human and machine workers are independent, on-demand, scalable, and distributed. For example, online workers and friends are delegated individually and on-demand, who may not know the existence of others. The structure of workers is "flat" with no explicit structure, which is different from traditional organization where humans form sophisticated structures, such as hierarchy. Similarly, machines work as individual/distributed units with no explicit structure. Therefore, human workers and machines can be delegated in a scalable way, due to their simple structure.

Besides, both human and machine workers need to receive well-defined subtasks that match their expertise/functionality. They do not have the whole context—such as the back-

ground and the requester's intent—they should be given clearly defined inputs and the requirements of outputs. More importantly, the subtasks should match the expertise of humans and the built-in functionality of machines.

While human and machine workers share the above commonality, they have some key differences. First, human workers need to consider cognitive issues (e.g., motivation, commitment), while machines do not. For example, friends/colleagues may be willing to help but may not be fully committed to your task, especially when they have own projects. Also, online workers can be motivated by paying incentives but their commitment is unpredictable. Second, for the same subtask, human workers may generate more diverse output than machines. This is because humans with diverse knowledge may interpret the subtasks differently, while machines strictly follow the given commands to generate stable output.

These differences indicate that workers have different *degree of determinism*: human workers have low determinism while machine workers have high determinism. Low determinism of human workers can be advantageous for creative tasks since more diverse ideas may be generated. However, it also implies that the outcome of delegation may not meet the intent of the requester. The uncertainty of performance needs to be mitigated such that the requester becomes more confident to delegate to those workers. On the other hand, high determinism of machine workers is valuable for automatable tasks since the output products are highly reliable. Nonetheless, it implies that the tasks that can be done by machines are fixed/limited. To achieve more flexibility, IoT machines are introduced so that machines can accomplish more complex tasks based on the sensor data and logic. However, those tasks are still pre-defined at the time of manufacturing, which cannot meet the diverse intent of requesters. Therefore, this limitation needs to be mitigated by allowing requesters (e.g., factory workers) to customize flexible tasks.

This dissertation aims to balance the low determinism of human workers to meet the intent of requesters. Besides, this dissertation aims to balance the high determinism of machine workers to meet the diverse needs of requesters.

To that end, this dissertation develops a series of strategies and prototypes. The relationship of the prototypes is shown in Figure 1.1. Specifically, *BlueSky* develops the most strict workflow for crowd workers who have the lowest degree of determinism. *CoStory* allows

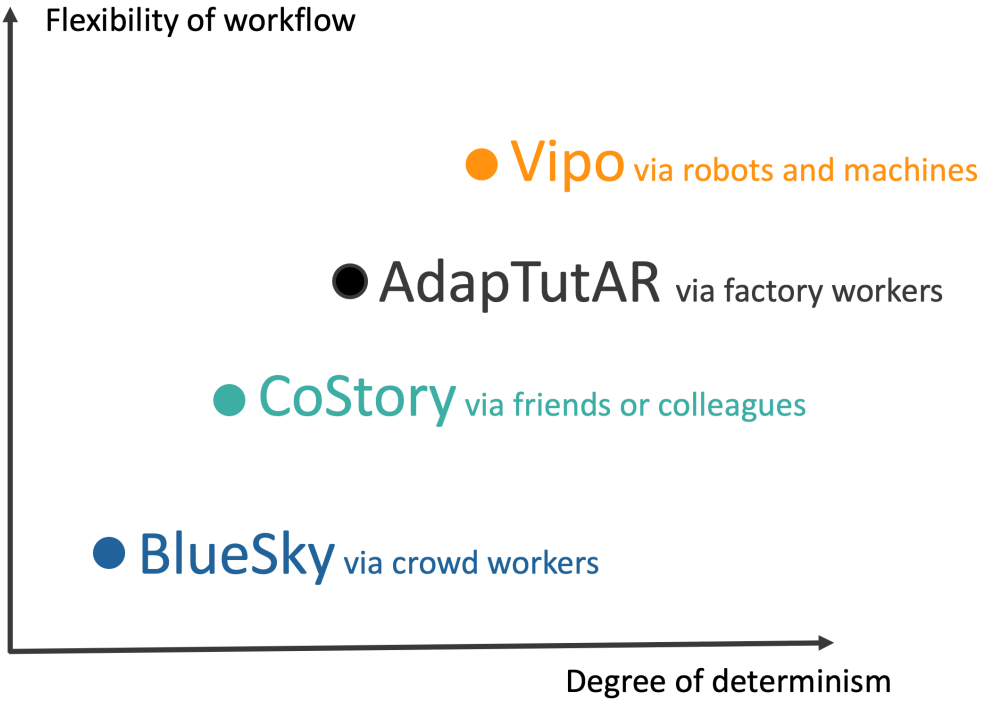

Figure 1.1. Both human and machine delegations rely on enforcing a workflow for workers. The flexibility of the workflow varies with the degree of determinism of workers. BlueSky enforces a strict workflow since crowd workers have low degree of determinism. CoStory has a more flexible workflow since friends or colleagues have higher degree of determinism than crowd workers. AdapTutAR enables an even more flexible workflow since factory workers have higher degree of determinism. Vipo enables the most flexible workflows since robots and machines have the highest degree of determinism.

requesters to follow a generic workflow to delegate tasks to friends or colleagues who have medium degree of determinism. Note that crowd workers are unknown and anonymous to the requesters, so they are considered as lower determinism than friends or colleagues who are known and familiar. *AdapTutAR* is a delegation/tutoring system for factory workers based on Augmented Reality (AR) and adaptive tutoring. When it comes to customizable routine tasks, factory workers are typically full-time employed with similar skills, and thus have higher determinism than friends or crowd workers. Finally, *Vipo* allows requesters to customize flexible workflows for robots and machines which have the highest determinism.

1.1.1 Creative Tasks with Crowdsourcing

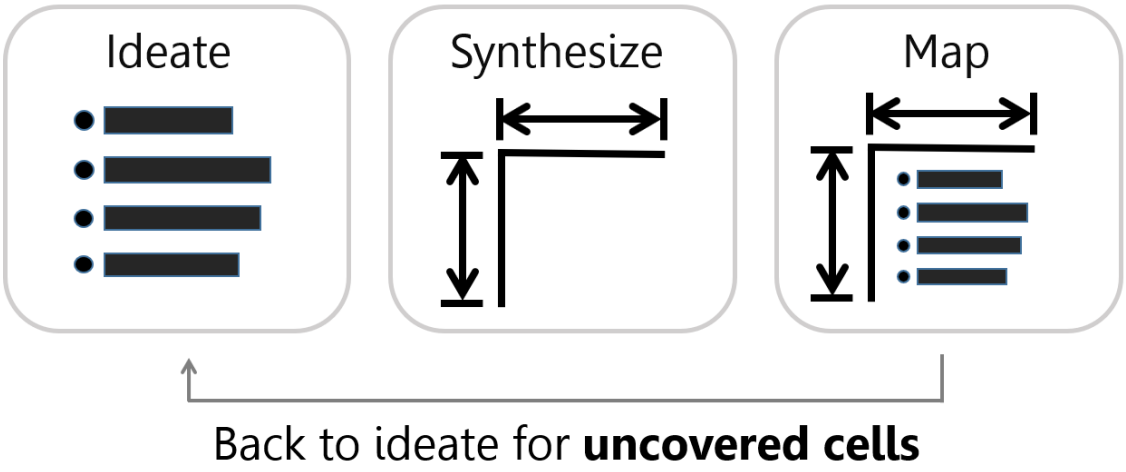

Figure 1.2. The ideate-synthesize-map workflow of BlueSky guides online workers to enumerate ideas with more specific steps.

BlueSky is a system that uses online workers to help enumerating ideas. The system engages online workers to enumerate a **uniform sample** of ideas in an idea space. For example, a possible scenario could be that a political campaign wants to prepare its candidate for the full range questions that might arise in a debate. By giving an input to BlueSky (e.g., "political debate question", "ways to use a brick", etc), accompanied by a brief explanation of the context, the system then coordinates workers on Amazon Mechanical Turk (AMT) to generate a list of ideas covering the whole idea space.

Although online workers have the potential to generate a rich set of ideas due to their diverse background and knowledge, many ideas would be deemed duplicate. An uncoordinated crowd effort would likely lead to some ideas clustered around those that are easiest to think of. To solve this problem, BlueSky introduces a workflow called *ideate-synthesize-map* (Figure 1.2). This workflow is a three-step process that first generates initial ideas, then builds an ontology of the ideas on-the-fly to categorize the ideas, and finally guides the workers to generate ideas for not-yet covered idea space. By enforcing this strict workflow, online workers are efficiently coordinated to enumerate ideas for a given topic.

1.1.2 Creative Tasks with Friendsourcing

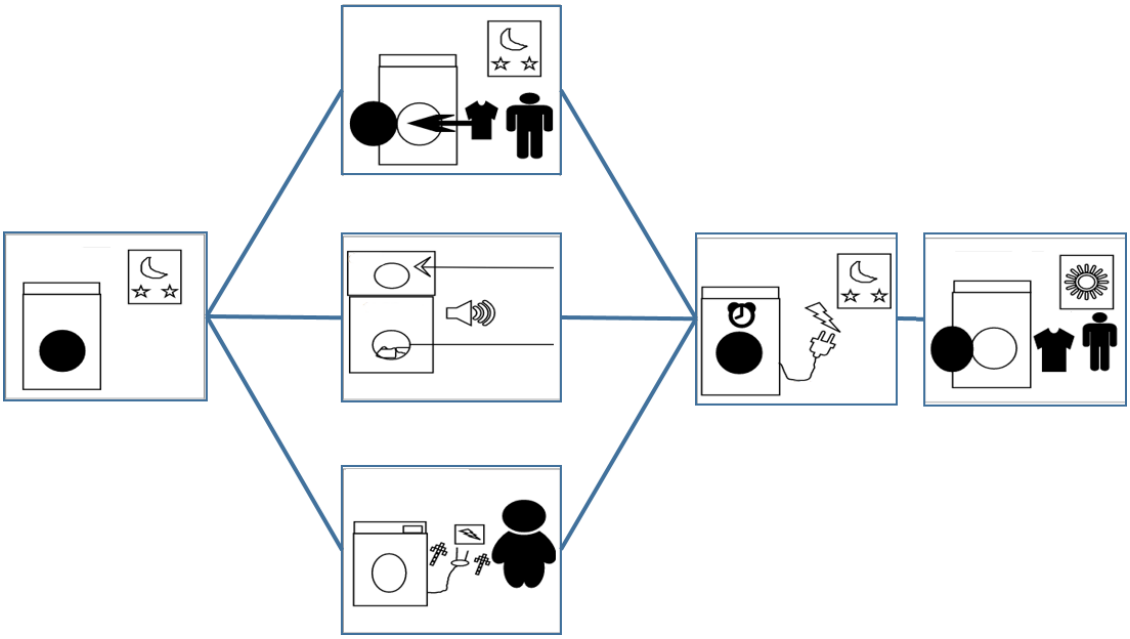

Figure 1.3. CoStory allows designers to request alternatives from contributors for key interactions. The panels above are part of a storyboard created by a participant in our user study. The key interaction is "how to setup the washer", which has three alternative designs.

CoStory is a system that enlists alternative designs via *friendsourcing*, such as friends and colleagues. For example, when confronted with challenging interaction design problems, CoStory allows requesters to easily get brief assistance from friendsourcing to bring complementary experiences and fresh perspectives.

Unlike within-team collaboration, these helpers are not expected to be familiar with the anticipated use context—much less the rationale for the design choices up to that point. CoStory presents a workflow, accompanied by a hierarchical storyboarding system, that allows requesters to decompose the design problems into simpler, self-contained subtasks and delegate just the part of the design problem that needs help. Furthermore, the alternative ideas generated by helpers are organized as tabs (similar to browser tabs), which enables efficient comparison and management.

The workflow of CoStory is less strict than that of BlueSky so that requesters in CoStory have more freedom to control which part to delegate and how much context to present to the helpers.

1.1.3 Machine Operation Tasks with Factory Workers

In modern manufacturing industry, workers are often assigned with new workflows and required to catch up quickly. Transferring the workflows to unfamiliar workers by experts is time consuming and expensive, which hinders the productivity. Using pre-recorded tutorials can make it more scalable, but the training process is less effective than that with human tutor. AdapTutAR is a system that enables efficient human-to-human machine task transfer. A requester (typically an expert) first records their motions and interaction with machines, and then workers can follow the tasks through AR headset. One core objective is to make the recorded tutorials adapt to different workers' performance and prior knowledge. To that end, I decompose the entire task into sequential steps, build an adaptation model to estimate the real-time status of workers, and adjust the tutoring content of each step according to the real-time state and history performance.

The workflow of AdapTutAR is more flexible than CoStory so that requesters in AdapTutAR can customize different tasks and delegate to factory workers.

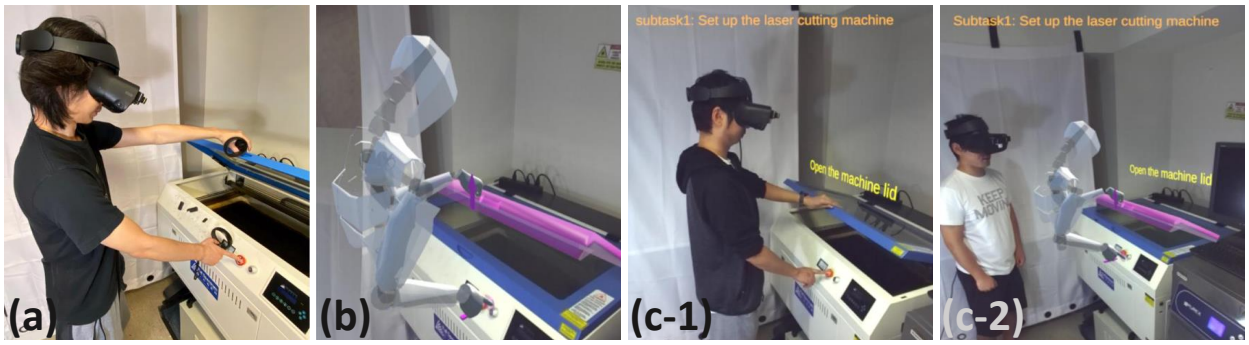

Figure 1.4. An overview of AdapTutAR workflow. (a) An expert records a tutorial. (b) The tutorial is represented as an avatar and animated components with arrows. The expert can edit the tutorial by adding *subtask* description and *expectation of step*. c) The same tutorial is adaptively shown to two learners. The learner in (c-1) is given less tutoring contents than the learner in (c-2) due to the difference of their experience and learning progress.

1.1.4 Flexible Workflows with Robots and IoT Machines

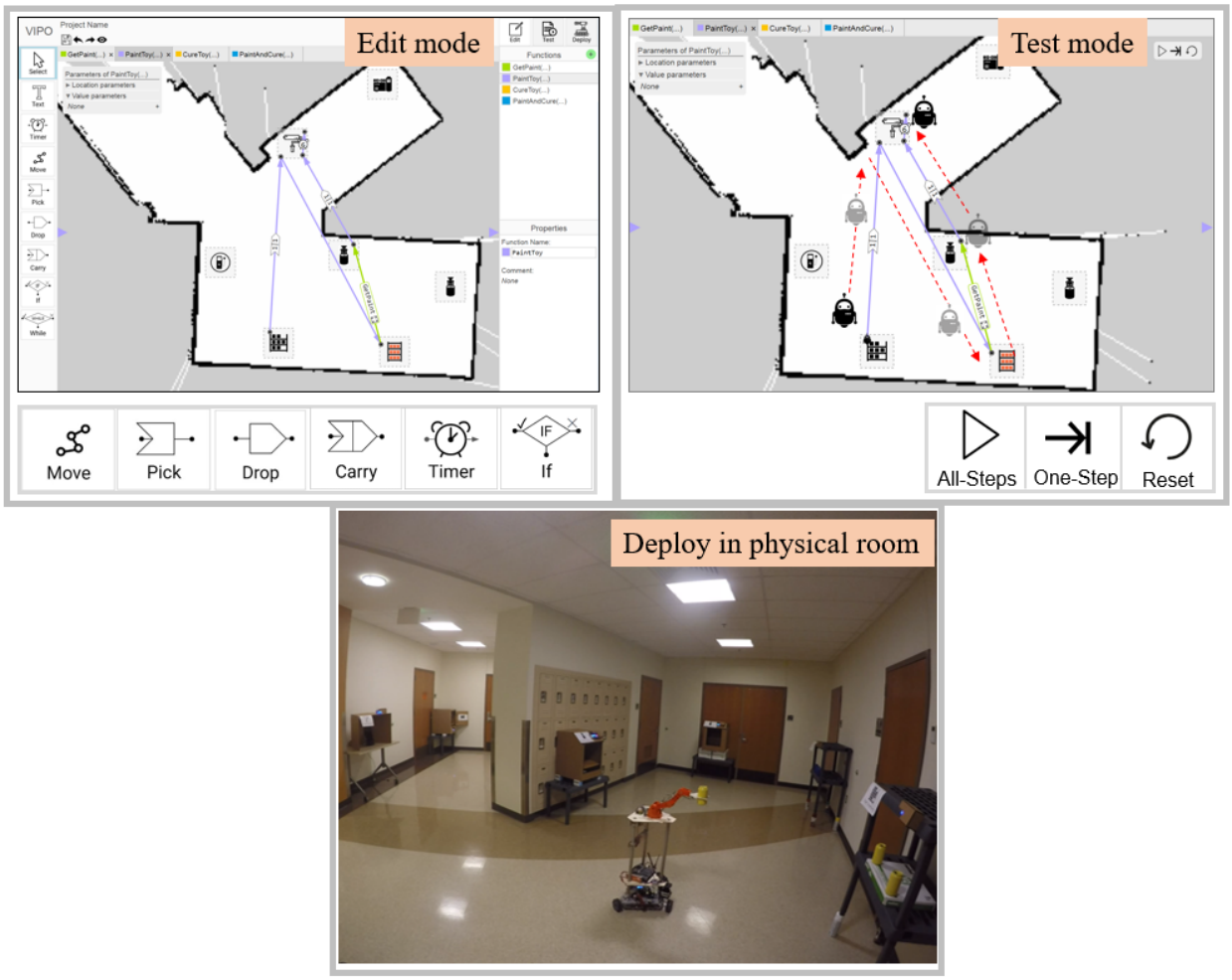

Figure 1.5. VIPO programs are created using standard programming constructs over a 2D floor map using the VIPO editor (top-left), then tested in simulation (top-right), and deployed to mobile robots that interact with IoT devices in the physical environment (bottom).

Vipo is a system that allows users to program flexible workflows for robots and IoT machines. For example, maintenance workers may want to program a mobile robot to pick the desired tools to the working zone. More interestingly, the workers may first program a 3D printer to print a replacement part, then ask the robot to pick it once the part is ready, then carry the part to a polishing machine to polish, then carry to the working zone, and finally notify the maintenance workers. Such workflows are highly customizable and dynamic, which

requires an efficient workflow programming tool that can be used by workers who are directly involved with a given manufacturing process.

Visual programming in the spatial context of the operating environment can enables mental models at a familiar level of abstraction. However, spatial-visual programming is still in its infancy; existing systems lack IoT integration and fundamental constructs, such as functions, that are essential for code reuse, encapsulation, or recursive algorithms. Vipo enhances with two significant capabilities. First, the Vipo language allows workers to write programs using *functions*. Second, the Vipo architecture integrates IoT devices into the programming and execution environments with no prior configuration.

1.2 Contributions

The core contribution of this dissertation is a set of insights and prototypes to simplify the task division and balance the variance of workers with different flexibility of workflows.

1. This dissertation introduces techniques to successfully delegate complex tasks to crowd workers. Crowd workers are unknown and anonymous to the requesters and thus considered as low degree of determinism. I present a system that decomposes the tasks into a three-step workflow that guides crowd workers to accomplish tasks that they could not succeed at normally.

2. This dissertation summarizes the pattern of *asymmetric collaboration* in the context of delegating tasks to friends or colleagues. By making use of the latent hierarchy of the tasks, requesters can easily decompose the tasks into meaningful subtasks that helpers just need to know.

3. This dissertation introduces an *adaptation model* that enables adaptive tutoring of pre-recorded AR tutorials for machine tasks involving spatial and body-coordinated interaction based on machine state and user activity recognition.

4. This dissertation introduces a *spatial-visual programming language and architecture* to easily program flexible and customizable workflows for robots and machines. Creating

a visual mapping of digital and physical environment can simplify the task planning, and increase the comprehensibility of the tasks.

1.3 Thesis Statement

This dissertation provides evidence that complex and creative tasks can be delegated to humans with low degree of determinism, and customizable routine tasks can be delegated to a hybrid of humans and machines with high degree of determinism. A common strategy is that the latent structure of the tasks is first identified, and is further used to guide the task division and system design. For example, the creative task in BlueSky (i.e., enumerating ideas) needs to build an ontology of the ideas to help guiding efforts toward the uncovered idea space. Similarly, the creative task in CoStory (i.e., soliciting alternative designs) relies on the fact that an interaction design can be expressed in different level of details. Thirdly, the machine operation task in AdapTutAR assumes that each operation is verifiable (i.e., being able to verify if the machine is set to an expected state) and order-dependent (i.e., following a specific order). Lastly, the automatable tasks in Vipo (e.g., manufacturing processes) can combine small and simple functions to build more complex functions, which forms a hierarchy.

The thesis statement is as follows:

> *Decomposing the latent structure of tasks can simplify task division and communication, and eventually enable efficient delegation.*

1.4 Dissertation Overview

To begin, Chapter 2 provides a literature review to the major research challenges in task delegation for humans and machines, especially in creative tasks and customizable routine manufacturing tasks.

The core of this dissertation introduces task delegation for different workers through dedicated systems: Chapter 3 introduces the BlueSky system, which demonstrates how to delegate creative tasks to crowd workers. Chapter 4 introduces the CoStory system, which demonstrates how to delegate creative tasks to friends or colleagues. Chapter 5 introduces

the AdapTutAR system, which demonstrates how to delegate customizable routine tasks to factory workers. Chapter 6 introduces the Vipo system, which demonstrates how factory workers can customize (routine) tasks for robots and machines.

The last chapter includes an overall conclusion of the dissertation.

2. RELATED WORK

This chapter discusses the prior work in task delegation, creativity methodologies and supporting tools, adaptive tutoring, and task planning for robots and Internet of things (IoT) machines.

Part of this chapter has adapted, updated, and rewritten content from a paper at Creativity & Cognition 2017 [7], a case study at CHI 2021 [8], a paper at CHI 2020 [9], a paper at CHI 2021 [10], and an unpublished paper (coauthored with Alexander J. Quinn). So in this chapter, all uses of "we", "our", and "us" refer to all the coauthors.

2.1 Task Delegation

The basic idea behind delegation is that *requesters* transfer tasks to *workers* when requesters cannot achieve their objectives alone. Due to the multidisciplinary nature of this problem, I draw on background from social science as well as computer science.

Delegation can take place in many situations. First, when people are absent, their jobs can be delegated to someone else. Such backup scenario is pervasive, such as when people are on a business trip or training. Second, administrative delegation happens when job functions are distributed from higher job positions to lower job positions in the organization structure [1], [2]. Third, delegation is useful in collaboration of work. People often need to collaborate with others in the same organization or other organizations, especially when they lack the knowledge or resources to achieve the objectives.

This dissertation focuses on the third case. In particular, we focus on collaboration with individuals or small organizations, rather than large organizations. This is because in large organizations, delegation also needs to consider many other issues, such as authority/permission control [11], [12], security, and intellectual property. Those issues are not the focal point of this dissertation.

Delegation can take many forms: human-to-human, human-to-machine, machine-to-machine, and perhaps even machine-to-human. In this dissertation, I focus on the human-to-human, and human-to-machine forms of delegation.

2.1.1 Task Delegation and Coordination

As mentioned above, this dissertation focuses on the collaboration/coordination of work. In organization theory, the term coordination is also paralleled with collaboration [13], co-operation [14], [15] and integration [2]. It is through coordination that *requesters* are able to engage *workers* effectively towards a common interest, since requesters typically lack the knowledge, capabilities, or resources to achieve their objectives alone.

The term coordination has various meanings in different contexts, such as in economics, computer science, organization theory, and biology. There appears to be no commonly accepted way of defining coordination. For example, Larsson [16] lists nineteen different definitions and Malone & Crowston [3] report on eleven definitions. For the purposes of this study, I have adopted one definition from [3] as: *"The joint efforts of independent communicating actors towards mutually defined goals"*. Here, since the coordination has a clear role of requesters, the mutually defined goals should be in align with the goals of requesters.

Coordination needs to resolve challenges in division of task and management of inter-dependencies [1], [2]. The task division challenge includes the consideration of how to divide subtasks in a meaningful and cohesive way so that the workers have enough context to work with. The managerial challenge includes the consideration of how to manage the interfaces between subtasks, how to match the subtasks with suitable workers, and what mechanisms are appropriate to achieve enough collaboration between workers.

Coordination is closely related to communication, which is considered as a source of establishing shared meanings. The importance of communication has been emphasized by many researchers [3], [4], [16], [17]. The key challenges of communication includes: *"How [...] can actors establish common languages that allow them to communicate in the first place?"* [3] and *"[...] the heaviest burdens are placed on the communications system by the less structured aspects of the organization's tasks, particularly by activity directed towards the explanation of problems that are not yet well defined."* [1].

2.1.2 Human-to-human Delegation

Tasks can be delegated to organizations or individuals. There is already great deal of theory about coordination within organizations or with other organizations [1], [2], [14]. This dissertation focuses on delegating to individuals, rather than delegating to an organization as a whole.

One option is to delegate work to one or more experts. For example, outsourcing software development tasks to external developers is a common practice to save costs. Also, platforms like Upwork[1] and 99designs[2] enable requesters to hire expert freelancers for various types of tasks, such as writing and logo design. This type of delegation is still on a formal basis, which acts similarly to delegate to an organization.

The second option is *crowdsourcing*, which is a practice of delegating tasks to a lot of crowd workers from online platforms like Amazon Mechanical Turk (AMT[3]). Jeff Howe [18] defines crowdsourcing as *"[...] the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call."*

While crowdsourcing is typically used to handle relatively simple tasks, such as image labeling and receipt transcription, there are many efforts trying to apply crowdsourcing to complex tasks. For example, CrowdForge [19] is a general purpose framework that provides a small set of task primitives (partition, map, and reduce) that can be combined and nested to address complex crowdsourcing problems. Besides, Soylent [20] is a word processing interface that enables writers to call on crowd workers to shorten, proofread, and otherwise edit parts of their documents on demand. Flash organizations [21] is a system in which crowd workers are structured like organizations to achieve complex and open-ended goals.

The third option is *friendsourcing*: delegating to a group of friends, colleagues, or other people available. Friendsourcing is a term introduced by Michael Bernstein, et al [22], [23], which is a technique to collect information or execute tasks in a social context by mobilizing a user's friends and colleagues. Specifically, they design systems to extract information about

[1]↑https://www.upwork.com/
[2]↑https://99designs.com
[3]↑https://www.mturk.com/

peoples' interests and preferences by encouraging friends to explicitly or implicitly share that information.

The fourth option is to delegate task to factory workers. Manufacturers around the world have worked on standardizing practices and making tasks less complex in an effort to keep their employees productive on the job. However, there are still many challenges of delegating tasks to workers in the trend of flexible and self-configuring production [5], [6]. In particular, workers often need to learn new production processes, due to the frequent production shifts and high worker turnover [24].

Moreover, in many industries, factory workers primarily operate machines to accomplish tasks. In this dissertation, I will focus on delegating machine operation tasks to factory workers. Nonetheless, an interesting perspective might be that delegating tasks to factory workers is essentially delegating to machines. In other words, it seems reasonable to consider machines as the end workers, while factory workers as the middle layer of the delegation. In a loose sense, this type of delegation may be confused as *human-to-machine* delegation that will be discussed later. Here, I still consider this type of delegation as human-to-human for two reasons. First, not all tasks delegated to factory workers are purely done by machines. Some tasks require substantial human effort and experience, such as welding. The human-to-machine delegation in the next section, however, assumes that automatable machines and robots are the primary operators. Second, this dissertation focuses on training workers to master machine operations, which is more related to cognitive and learning aspect of humans.

This dissertation focuses on the last three options of human-to-human delegation. While some general-purpose frameworks have been mentioned above, they can not be directly applied to the task types in this dissertation. Therefore, I introduce different techniques and frameworks to enable delegating these tasks. The later section will cover more details of the task types as well as the related work in handling them.

2.1.3 Human-to-machine Delegation

Humans routinely delegate tasks to machines. The list of machines is endless, such as rice cooker, printer, laser cutter, and so on. Most machines are primarily designed to work alone, without interaction with other machines.

A system of interconnected machines, however, has the potential to achieve more complex, flexible tasks that single machines cannot achieve alone. Internet of things (IoT), for example, represents a system of interrelated computing devices, mechanical and digital machines that are provided with many features, such as the ability to transfer data over a network. These features can increase the flexibility and diversity of potential applications of the IoT machines [25].

As mobile robots and human workers become more tightly integrated within IoT environments, the task of instructing the *Internet of robotic things* has become increasingly complex [26]–[28]. For example, in manufacturing factories, it becomes challenging to program workflows of mobile robots delivering parts and inter-operating with manufacturing equipment. As manufacturing processes increasingly depend on customization and product changes, the effort needed to create or modify workflows becomes a bottleneck. Furthermore, some responsibility for programming robots and their interactions with IoT devices must shift to the workers directly involved with a given manufacturing process [29].

The later section will cover more details of the related work in coordinating robots and IoT machines.

2.2 Creative Tasks and Methodologies

Creativity is a common objective of task delegation. Different types of creative tasks have intrinsic characteristics that a generic methodology may not be able to address well. This section discusses the methodologies and tools for two types of creative tasks: idea enumeration and interaction design.

2.2.1 Idea Generation and Enumeration

Creative tasks can be made less dependent on chance by guiding the process with some element of the idea space. Yu et al. have shown that crowds can find analogies from within a large set of ideas [30], and that such analogies can provide the seed for new and valuable ideas [31].

Recent work relating to the enumeration of unbounded sets with crowds has established a foundation for our work [32], [33]. However, enumerating creative idea spaces is fundamentally different from drawing ideas from the Internet at large.

Dimension-driven ideation

Dimensions and values have been used for ideation of graphical designs, by individuals and crowds. Talton et al. demonstrated one of the first interfaces that allow individual users to explore an idea space by manipulating sliders to control various dimensions [34]. A similar strategy is embodied by Attribit, which also engages workers on Mechanical Turk to help describe relationships between resulting objects (e.g., *"The right part makes the shape look ___ than the left part."*) [35].

Quasi-comprehensive enumeration of an idea space

A recent effort enumerates ideas for inventions using machine computation. The website's stated goal is *"to algorithmically create and publicly publish all possible new prior art, thereby making the published concepts not patentable"* (http://allpriorart.com). Here is an example:

> *"Devices and methods for therapeutic photodynamic modulation of neural function in a human. Product gas in the vapor phase is drawn from the head space above the liquid level and condensed to form the product fuel. The adapter segment is positioned at the first end and is configured to be coupled to another component."*

All have been computationally generated using algorithms that are not yet publicly disclosed.

Morphological Concept Generation

Morphological concept generation is a method from engineering design that involves mapping an ideas space in a way that could potentially be used to enumerate the ideas. It generates concepts based on a morphological chart. A morphological chart in essence is a table of *functions* (desired capabilities for the object being designed) and an array of *means* (ways to implement or achieve those capabilities) [36]. The space of possible designs can then be viewed as the combination of available means for each function [37]. Exploring that space of combinations is a popular method of engineering design [38]–[41]. Web-based morphological charts have been demonstrated, including support for functional analysis, conception generation, and concept evaluation [42]. Since morphological charts are typically used in a large idea space, they can naturally lead to combinations of features that would make the design impractical or nonsensical. Compatibility matrices can be used to evaluate whether functions can reasonably be combined [38], [43].

Creativity Support Tools

Key ideas from creativity support tools [44] have been applied to crowdsourcing to amplify the potential for creative output. Crowds inherit previous examples and combine atypical features to produce more creative ideas [45], [46]. Challenges, however, stem from the vague requirements for discovery and innovation, as well as from the unorthodox user behaviors and unclear measures of success [44].

2.2.2 Interaction Design

Interaction design is about designing interactive products that people communicate and interact with. Interaction design is widespread in many fields, such as designing physical products and software interfaces. Also, it is widely performed by designers with a variety of design expertise for different purposes, ranging from professional designers for commercial products, amateur designers (e.g., students) with limited design experience for design-oriented projects.

The four stages of interaction design involve establishing requirements, designing alternatives, prototyping, and evaluating [47]. In the phase of designing alternatives, designers need to consider many aspects of users. At the very least, they need to determine the target users, and understand the potential interactions with the products in different scenarios [47]. Having a better understanding of target users and potential interactions can help design products that will fit those niches and reduce the cost by avoiding implementing inappropriate designs.

To that end, collaboration is often used to generate creative and diverse design alternatives. Bringing together people with different expertise and perspectives can help mitigate the design fixation [48], [49]. Collaboration can happen not only within design teams, but also between people peripheral to the design teams, such as colleagues and friends. In this dissertation, we focus on the latter collaboration pattern in which designers enlist assistance from people with only limited interest, responsibility, and awareness of the design problem.

Authoring and Communication Methods

Various design methods can be used to enable designers to author and communicate the design context. Atasoy and Martens [50] made a summary of common methods, such as mind mapping, affinity diagramming, scenarios, storytelling, sketching, and storyboarding. Specifically, scenarios, storytelling, and storyboarding may be more suitable for interaction design than other methods due to their abilities to describe the target users and the potential interactions in a story that people can easily relate to. Moreover, unlike scenarios and storytelling, storyboarding tells the story in a visual way, and thus can act as a lingua franca in the communication between design teams [51]. Therefore, we choose storyboard as the method in this dissertation.

Storyboarding is a visual storytelling method represented as a series of pictures or sketches. Storyboarding is commonly used in many fields, such as film making, animation planning [52], and comic books [53]. For example, a storyboard for animation can give animators a better idea of how a scene will look and feel with motion and timing. Key frames of the animation can be represented as storyboard *panels*.

When it comes to interaction design, storyboards can be used to "illustrate the design concept in the context of users, task, and environment" [54]. Storyboards can be used as *low-fidelity prototypes* to foresee key interactions that users may engage with the products [47]. Additionally, storyboards may implicitly or explicitly express the time passing of the interactions, which may be helpful for designers to consider the potential challenges that users might face over time [55].

When many storyboards are created to represent different design alternatives, managing the complexity becomes critical for the communication and collaboration. In this dissertation, we use a hierarchical storyboard to manage the complexity while maintain its sequential appearance.

2.3 Adaptive Tutoring

This section discusses prior approaches to adaptive tutoring for general contexts, and the sources and targets of adaptation for AR/VR specific tutoring systems. These works inform the system design and features of AdapTutAR.

2.3.1 General Strategies for Adaptive Tutoring

In human-based tutoring systems, instructors perform a multi-dimensional role, from providing classroom instructions, providing feedback to trainees (questioning, suggesting hints or direct orders) and even varying the difficulty of the training to suit the trainee [56]. Adaptive instructional systems aim to replicate these roles in the absence of a human tutor. These computer-based systems guide learning experiences by tailoring instruction and recommendations based on the goals, needs and preferences of individual learners in the context of domain learning objectives [57]. Thus, the goal of adaptive tutoring is not just to facilitate, but optimize the learning, retention and transfer of skills for users between the training and real-world environment.

Instructional strategies for adaptive tutoring can be grouped into two general approaches: macro-adaptive and micro-adaptive [58], [59]. Macro-adaptive approaches provide adaptation based on metrics collected prior to training. Generally, they use metrics such as learner

preference and experience to establish methods for individualized task selection or content difficulty [60]. Micro-adaptive use real-time metrics to provide adaptations in a dynamic fashion. They perform adaptations during the training using factors such as user performance, behavior and errors to provide guidance and feedback [61], [62]. However, to determine which adaptive approach should be used, it is useful to understand the possible sources and targets of adaptation.

Sources of adaptation pertain to factors which cause or trigger the adaptation to occur. They largely stem from learner-based metrics such as individual performance, working memory capacity [63], prior expertise [64], learning preference and traits [62]. Intelligent tutoring systems usually employ a learner model [65], [66] which collects learner data to ascertain their knowledge state, recognize errors and generate adaptations. ***Targets*** of adaptation represent those instructional components which actively change based on the source of adaptation. Broadly, these targets can include the feedback, visual representation of information, sequence of workflow, learning pace and others [62]. Effective and personalized feedback is important for learning due to the different characteristics of users, as shown in the works by Gutierrez and Atkinson [67] and Bimba et al.[68]. Alternatively, Brusilovsky and Su [69] explored the relation between adaptive visualization and learner knowledge levels, while Beyyoudh et al. [70] focused on providing the optimal sequence of pedagogical steps.

The sources and targets of adaptation define the general adaptive approach of the tutoring. While AdapTutAR uses a combination of macro and micro-adaptive approaches for adaptation, the following sections discuss the different sources and targets of adaptation specific to the AR/VR context.

2.3.2 Source of Adaptation in AR/VR

AR/VR applications have shown benefits to communication and learning by displaying effective and adaptive information that enhances users' understanding of subjects. While adaptation within the AR/VR context has not been studied extensively, some sources of adaptation in AR/VR tutoring systems can include learner performance, expertise, behavior (gaze, distraction, emotion), task-type and spatial location. From the previous section it

can be seen that generic tutoring systems largely monitor user-based factors. In the case of AR/VR, these factors can be classified into two groups: User and Environment [71].

User refers to the person using the application. User's performance is widely used in directing the tutoring workflow, initiating appropriate feedback [56], [72], [73], or in selecting macro and micro-adaptive strategies [74]. Fender et al. leveraged user behavior and object position to adapt the position [75] and size [76] of AR displays. Learner gaze can be used as a measure of transparency [77], allowing the tutoring agent to make inferences about the learner confidence, whether or not guidance is necessary and what they are likely to do next [64]. Finally, Rodenburg et al. elaborate on the correlation between learner expertise and level of fidelity in simulation based tutoring environments [78].

Environment refers to the physical context where users are interacting. When dealing with machine environments, Cao et al. [79] in their exploratory study categorize the steps of *machine task*s into three types depending on the physical actions performed: *local, spatial* and *body-coordinated* tasks. Their user study suggests that users prefer different visual abstractions of the AR avatar tutors depending on the task type. Lages and Bowman [71] used information about physical surroundings and relative positions of the environment layout to focus on position-adaptation. Additionally, Herbert et al. [80] use spatial 3-D information from the real-world to detect errors, provide feedback and sequence tasks.

By taking information from the user and environment into account, AdapTutAR generates adaptive tutorials for *machine task*s. The next section describes the various targets of adaptation in AR/VR for *machine task*s.

2.3.3 Target of Adaptation in AR/VR for *Machine Task*s

*Machine task*s can be defined as a sequence of physical and spatial operations involving machines in a production environment [79]. Some examples of AR-based tutoring for machine environments include usage of industrial machinery [81]–[83], facility monitoring [84], [85] and vehicle maintenance [86], [87]. Considering that most *machine task* operations involve human motion, the targets of adaptation must focus on using the right type of visualization content for the tutoring and the level of detail.

AR/VR visualizations include the usage of annotations, animated components and virtual avatars. While annotations [80], [88], [89] and animated components [90]–[93] have been used extensively in prior AR research to adapt feedback and guide users, avatars have been used to provide effective feedback for learning tasks that primarily require human motion. For example, Tai-Chi training platforms where learners learn from virtual coaches have been researched extensively [94], [95]. Cao et al. suggested the use of avatar as an additional instructional mode [79] after exploring the presence of avatar for tutoring *machine task*s. Similarly, Piumsomboon et al. studied the presence of an adaptive avatar to facilitate the collaboration between a local AR user and a remote VR user [96], [97]. Recently, Loki [98], a bi-directional mixed-reality telepresence system for teaching physical tasks, used the avatar to represent status of the learner and instructor in different physical spaces. By offering customized feedback from the avatar, users gain deeper understanding within synchronous learning.

Level of Detail (LoD) relates to the questions of *when,* and *how* much information should be presented to the user for optimal tutoring. Lindlbauer et al. [99] report an optimization approach leveraging cognitive load and the task environment to adapt MR interfaces to fit the user's context. Wegerich and Rötting [100] outline a context-aware adaptation system for spatial AR with the goal of displaying unambiguous information at the right time to the user, based on user attributes such as position and perception. For AR browsers, Tatzgern et al. [101] presented an adaptive information density display which used a level-of-detail structure to balance information against potential clutter on the display.

Prior works use these targets of adaptation to provide users with optimal amount of information for tutoring. The different sources of adaptation reviewed in the previous section are linked to various targets of adaptation in this dissertation. AdapTutAR significantly expands on these targets and presents an adaptation model that targets the optimal level of detail based on a combination of user and environmental sources.

2.4 Programming Tasks for Robot and Machines

This section discusses the prior work of delegating tasks to robots and IoT machines, including visual programming languages and interfaces, spatial-visual programming, and communication protocols between smart devices.

2.4.1 Visual Programming for Robots and IoT Devices

Visual programming interfaces make programming more approachable for non-experts and thus enable workers to author workflows for robots and/or IoT devices. Many visual programming interfaces have been developed to program tasks for IoT devices [102]–[104], robots [105]–[111], or for both robots and IoT devices [112]–[115].

These interfaces are mainly built on two authoring approaches: *form-filling* and *visual programming languages*. Form-filling approach allows users to fill a predefined form by adding actions or triggers via drop-down menus [112], [116], [117]. On the other hand, visual programming languages provide visual constructs (e.g., functions, conditions, and loops) to wire the sensory data and actions of robots or IoT devices into tasks, such as Blockly [118]. Since form-filling is less flexible than visual programming languages in authoring dynamic and complex workflows, most interfaces mentioned above have adapted or designed visual programming languages to author workflows in various formats, such as blocks [107], [118], data-flow [108], flow-chart [102], event-based, [111], or state-flow [113].

Although these visual programming languages could represent workflows in many formats, they lack suitable visual notations to represent the activities occurring within a spatial environment, such as delivering parts by mobile robots and interacting with machines. We design Vipo to provide users with handy notations to program workflows while maintaining the power of a programming language in the spatial domain.

2.4.2 Spatial-Visual Programming

The ability to sense the spatial relationship between objects and environments is one of the key benefits of programming via augmented reality (e.g., V.Ra [119]) or virtual reality

(e.g., Ivy [103]). Such benefit is automatically gained when authoring in 3D space, but is less obvious to obtain in 2D space. Given that users are more familiar with authoring via 2D interface and have easier access to 2D authoring tools (e.g., computers), it would be valuable to embrace the benefit of spatial-awareness in a 2D space.

Spatial-visual programming in 2D space is still in its infancy. A few research projects have explored this area. For example, Vizir [120] allows air traffic controllers (ATC) to author automation on top of a geographic airport map with a set of ATC-specific visual constructs. By placing the visual constructs above the map, Vizir tries to maximize the closeness of the control and actions, as well as the predictability of the automation. In addition, Ruru [109] designed spatial metaphors for input sensors to allow the position and orientation of an input relative to a device to be expressed visually. Kitty [121] allows users to sketch animated drawings to illustrate the spatial and temporal relationship between entities.

Although these approaches enabled users to create simple workflows or illustrations, they did not support *functions* that are essential for a programming language. Functions are supported in most textual languages and non-spatial visual programming languages mentioned above. Given that manufacturing workflows have the tendency to become more complex and customizable [29], functions can play an important role due to its reusability, modularity, and flexibility. In this dissertation, we designed and implemented functions in a spatial space. Along with other spatial constructs, the Vipo language aims to build an accurate conceptual model of the spatial relationship between programmed tasks and the environment.

2.4.3 IoT Protocols & ROS

IoT protocol is one of the key components to bridge the digital programming interface and the physical execution environment. Specifically, to program a workflow, users need to have access to the capabilities and sensory data of robots and IoT devices. In addition, these capabilities and sensory data should be represented in a format that is understandable by users.

Many protocols have been proposed to facilitate communication between connected systems [122]–[126]. For example, MQTT [123] is a publish-subscribe messaging protocol that is suitable for mobile applications. In addition, several protocols have been created to describe data/message in specific formats. For example, Resource Description Framework (RDF) [127] is a standard model for data interchange on the Web, while IoT-Lite [128] is a variation of RDF to describe IoT resources, entities and services.

This dissertation takes advantage of the Publisher/Subscriber functionality provided by Robot Operating System (ROS) [129] to enable status sharing and task coordination. Furthermore, this dissertation adapts the RDF protocol to describe the status and capabilities in a way without prior configuration.

3. BLUESKY: CREATIVE TASKS WITH CROWDSOURCING

This chapter presents the first example of human-to-human task delegation, which is delegating creative tasks to crowd workers. This chapter has adapted, updated, and rewritten content from a paper at Creativity & Cognition 2017 [7] and a case study at CHI 2021 [8]. All uses of "we", "our", and "us" in this chapter refer to coauthors of the aforementioned paper.

Design contests and group creativity support systems have demonstrated the value of crowds for producing a solution to a design or engineering problem. However, when the goal is not one idea, but many ideas, an uncoordinated crowd effort would likely lead to a set of ideas clustered around those that are easiest to think of. We present *BlueSky*, a crowd-powered system that coordinates microtask workers to enumerate a uniform sample of textual ideas on a given topic. Through the process of soliciting ideas, an ontology is developed, which is used to categorize the ideas in the list. That categorization then reveals combinations of attributes that have not yet been covered. Our evaluation with four list topics compared BlueSky to freeform solicitation of ideas with respect to comprehensiveness (coverage over the entire idea space) and uniformity (mitigating the tendency to emphasize ideas that are easy to think of).

3.1 Motivation and Contributions

With the rise of crowdsourcing came a recognition of its value of a distributed workforce for creative applications [45], [130], [131]. Web sites such as 99designs and InnoCentive have had success engaging crowds on design and engineering problems through design contests. By posting many such contests on one site, would-be contributors can match their individual knowledge and experience with problems to which they are well-suited. With a sufficiently large and diverse set of participants, the strategy can lead to a pool of candidate ideas from which a single winner is chosen.

Design contests—and *idea contests*, in general—depend on chance. Finding a great idea requires that among the people who view the contest, at least one will have just the right combination of inspiration or background knowledge to lead them to a winner [132]. Most

other forms of group creativity support tools are also primarily participant-directed, and thus effectively dependent on chance [44]. Moreover, these assume that the motivating goal is to find a *winner*.

This work engages crowds differently, in the form of mechanized labor—or, more aptly, mechanized creativity—to achieve a broader goal: *Enumerate a uniform sample of ideas in an idea space.* For example, a few examples of textual ideas spaces (i.e., ideas expressed as text) are below:

1. The developer of a new virtual reality system wants to understand the full range of possible applications.

2. A political campaign wants to prepare its candidate for the full range of questions that might arise in a debate.

3. A marketing firm wants to test a large and diverse set of text ad blurbs in an online experiment.

A truly exhaustive listing of *all* ideas for such topics would of course be impossible to produce. Many items on such a listing would be deemed duplicate, or at least "equivalent" by any reasonable standard. For example, given the prompt, "Ways to use a brick," the ideas "break a window" and "throw a brick through a glass pane" would probably be deemed equivalent. However, "break a window" and "crack a window" would depend on what one defines as *distinct*. Given such a definition, then a comprehensive *sample* could be achieved.

One possible standard for distinctness is suggested by legal theories of originality in the United States, which demand at least a modicum of creative thought [133]. Purely mechanical transformations of existing knowledge are deemed unoriginal (and thus ineligible for copyright). We take this a step further by requiring that for a new idea to be considered *distinct*, it must add creative value relative to the needs of the requester who solicits the list of ideas.

Even then, there could still be many possible definitions of "distinct" for a given ideation task. The choice depends on the intended purpose of the list. For example, a building supply marketer might treat two ideas as effectively equivalent if they would appeal to the

same segment of customers, or are used in a different setting. In contrast, a sculpture artist might differentiate based on the physical characteristics of the brick that are exploited. More concretely, we consider two ideas distinct if they differ with respect to any dimension that is deemed important by the requester. For example, "break a window" and "break a vase" differ by the dimension, "target of action" (e.g., window vs. vase).

Ideally, the definition of distinctness would come from the *requester* for whom the list of ideas is being created (e.g., the developer of the virtual reality system). That way, the kinds of variations could be aligned with what they need and value. One could imagine a system in which requesters actively participate, guiding crowd workers toward the avenues of variation that will produce the most value for that requester. However, for this research, we will aim for a process that is as self-contained as possible.

> **Objective:** Enumerate a sample of ideas spanning an idea space that are distinct with respect to some set of k dimensions.

This paper presents BlueSky, a system we developed to demonstrate a novel method for enumerating idea spaces (hereafter referred to as the *BlueSky method*). The inputs consist of a noun phrase describing the list topic (e.g., "political debate questions," "ways to use a brick," etc.), accompanied by a brief explanation of the context, to help workers understand. BlueSky then coordinates workers on Amazon Mechanical Turk (AMT), together with intermittent feedback from the requester, to generate a list of ideas covering every meaningful combination of values on the dimensions.

A *dimension* consists of a label and a set of values that can be used to partition the ideas in a list. Some dimensions are inherently discrete and finite (e.g., "i*s destructive*" $\Rightarrow$ {"*yes*", "*no*"}). The others will be clustered into bins or representative values (e.g., "*location*" $\Rightarrow$ {"i*ndoor*", "*outdoor*", "*underground*", "*outerspac*e"}).

Any item in an idea space can be mapped to a set of values for each of the dimensions. Two ideas that map to the same set of values are treated as equivalent. One of the primary goals of BlueSky is *comprehensiveness*: covering every compatible combination of values. The other primary goal is *uniformity*: controlling redundancy so that (ideally) only one idea

is acquired for each combination of values. As we will see, perfect uniformity is hard to achieve given the constraints of Amazon Mechanical Turk, the platform which we have used for this research. However, our evaluation demonstrates that BlueSky produces a much more uniform list of ideas (i.e., less redundancy) than freeform solicitation of ideas.

3.1.1 Contributions

The key contributions can be summarized as follows:

1. We show how *mechanized creativity*—driving the initiative and discovery from the system—can engage the creative intellect of crowd workers with less dependence on chance than unconstrained idea solicitation.

2. BlueSky, a system we developed, applies mechanized creativity to enumerate a comprehensive sample of an idea space efficiently.

3. Our evaluation demonstrates that a uniform and diverse sample can be more efficiently acquired by synthesizing a set of dimensions from an initial batch of unconstrained ideas and using those dimensions to guide further ideation—versus collecting the same number of ideas without constraints.

3.2 Initial Approaches

The central strategy of BlueSky is to discover the important dimensions and then continue soliciting ideas until all compatible combinations of dimension-values have been covered. However, developing this strategy into a working method and system was an iterative process.

In this section, we detail some of our early approaches. These illustrate the design challenges, which had to be overcome in order to develop our high-level strategy into the eventual BlueSky method and implementation, which will be presented in the following section.

3.2.1 Voluntary Hints Pointing Solution Path

To explore a wider idea space, our first strategy was to let workers ideate and voluntarily share the promising solution paths with future workers (Fig. 3.1). Participants were asked to enter ideas and rate existing ideas generated by others. While typing, they might see some similar ideas generated by others and be suggested to avoid duplicate ones. Once got stuck, they could click the "See some hints" button to acquire some hints which were shared voluntarily by other participants who had multiple solution paths.

We evaluated it with a specific list topic—"things that humans can do, but computers cannot"—and got 191 ideas from 12 distinct AMT workers. We found that the strategy of voluntary hints suffered from two main problems:

1. **Freeform ideation may lead to many redundant ideas.** Workers received no strict/tight constraint in the whole process of ideation. Instead, they were suggested, not required, to think a different idea once their ideas were duplicated with existing ones. There were only 167 distinct ideas (out of 191 in total) after de-duplication; meanwhile, a lot of them are similar. For instance, "make food" and "cook" are ba-

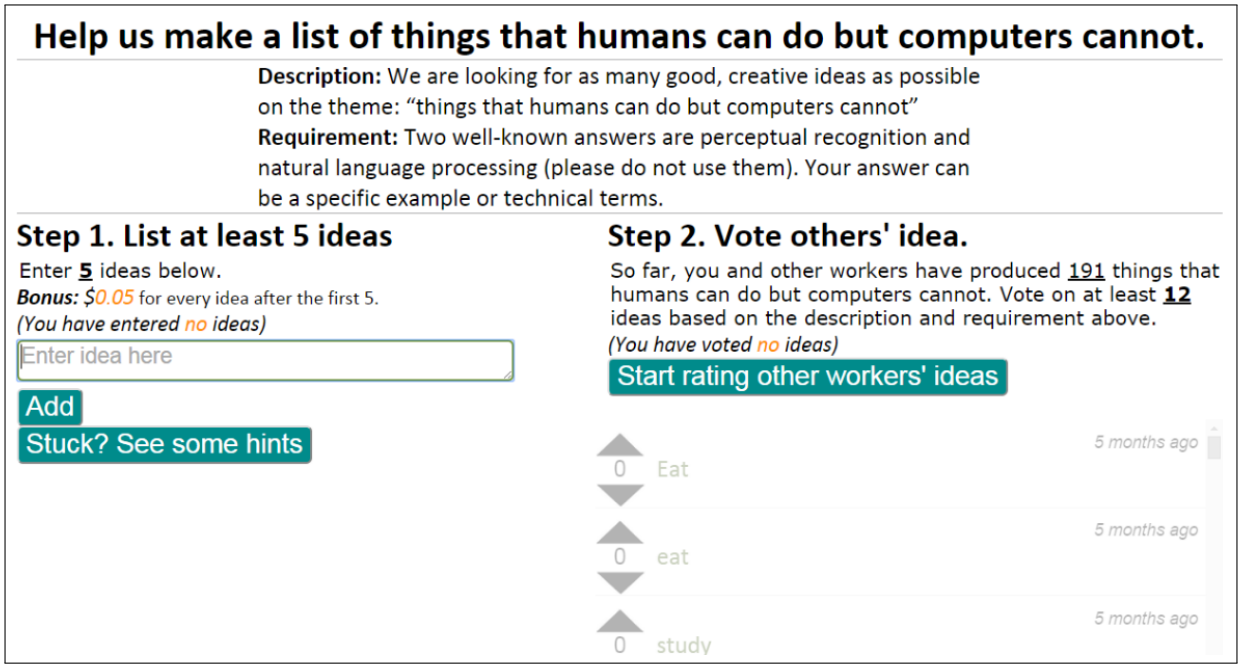

Figure 3.1. Voluntary hint interface with ideation (left) and voting (right).

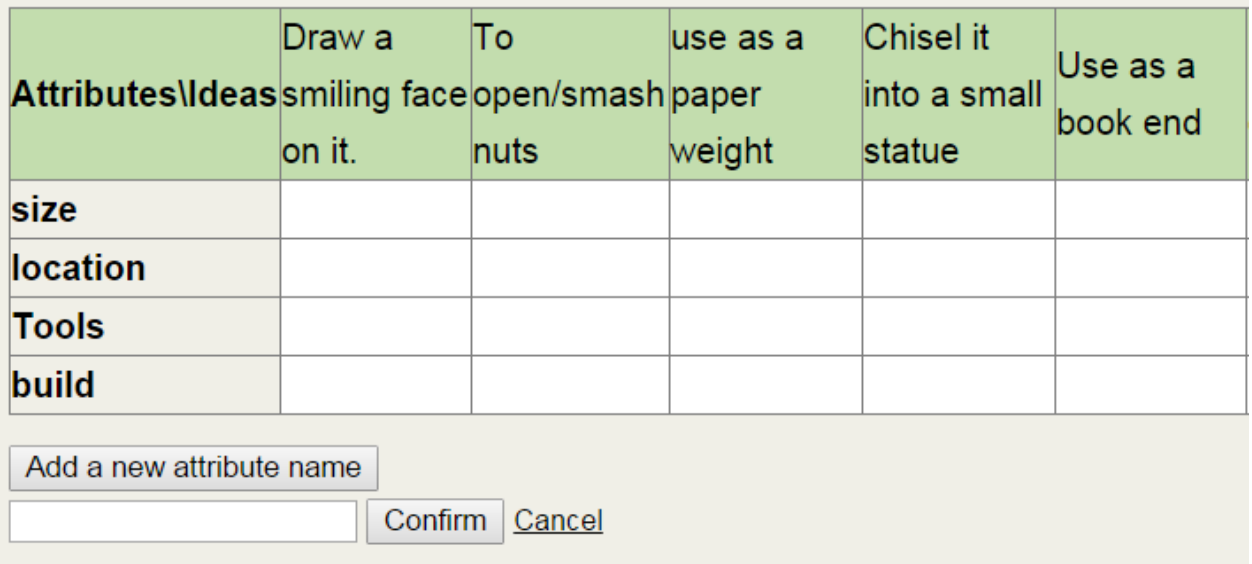

Figure 3.2. Attribute taxonomy interface. The column headers show the ideas and the row headers show the attributes.

sically the same; likewise, "play", "play go" and "play sports" are very similar. The possible reason is that workers often think in their familiar ways on the lack of compulsory constraint.

2. **Hoping to obtain hints from workers may be inefficient and infeasible.** During the whole process of ideation, no hint was added by workers. One reason could be that the task was so loosely restricted that workers might feel unnecessary to provide hints for next workers. We further realized that workers would not provide hints even in a tougher task but simply finish the task by their own.

We also observed creativity fixation effects [134]. Therefore, we decided to 1) add *constraints* to achieve less duplication and 2) reduce the exposure of others' ideas in the ideation session (e.g., removing the voting section and auto-suggestion).

3.2.2 Attribute Taxonomy

The second strategy was to create a taxonomy of *attributes* that are contained in most ideas. Figure 3.2 shows the interface that allows workers to propose new attributes and

extract details from ideas for each attribute. After the extraction, workers should merge the details into several values that could form a parent-child relationship with each attribute.

We tested it with four graduate students and one undergraduate student by giving a list of ideas for the topic "ways to use a brick". Under face-to-face environment, they could ask questions if they were confused about the process.

Similar to Cascade [135], we observed that they often gave overly broad attribute names such as "build" or "tools". Besides, the details they extracted were often in different levels of abstraction. Take attribute "build" for example, participants extracted both higher-level details like "something" and lower-level details like "house" and "pond for fish", from the ideas "build a house", "build a pond for fish", and "use to build something". This is because the ideas contained different levels of details. Furthermore, participants had great difficulty in merging those details into values. We thus decided to ask workers to provide values directly, instead of merging from details with unpredictable, different levels.

3.3 Definitions

Before describing the system design, we define some key terms, as they are used in this paper.

A **dimension** (d) is comprised of a finite set of **values** (V). We denote this relationship as $d \Rightarrow V$. Some dimensions are inherently discrete and finite (e.g., "is *destructive*" $\Rightarrow$ {"*yes*", "*no*"}). For others, the values represent clusters. For example, given the dimension *location*, there could potentially be an infinite number of locations in the universe, but for our purposes, the set of values will always be a finite set (e.g., "*location*" $\Rightarrow$ {"*indoor*", "*outdoor*", "*underground*", "*outerspace*"}).

Together, a set of dimensions form the basis of an **idea space**. As in linear algebra, this basis is not unique.

We use the term *dimension*, not *category*, because a dimension is intended to cover most (or even all) of the ideas (also referred to as *list items*) while a category may only cover partially. Also, in early testing, we found that some people interpret *category* like

dimension, while others interpret it like value. The terms dimension and value, while perhaps less intuitive, do not suffer from this ambiguity.

To chart an idea space, the values of a dimension should meet the following criteria as much as possible:

- **Orthogonal** (no overlap between each other). For example, "*occurrence time*" $\Rightarrow$ {"i*n January*", "i*n February*", ...} are orthogonal, but {"i*n January*", "i*n the spring*", ...} are not, because "January" has overlap with "spring".
- **Evenly-spaced** (nearly the same differences between each other). For example, "*occurrence time*" $\Rightarrow$ {"i*n January*", "i*n February*", "i*n March*", ...} are evenly-spaced, but {"i*n January*", "i*n June*", "i*n July*", ...} are not.
- **Complete** (all representative aspects). For example, *"color"* $\Rightarrow$ {*"red", "green", "blue"*} and {"*cold*", "*warm*"} are both complete, but {"*red*", "*green*"} are not.

3.3.1 Idea Space and Cell

In theory, every idea space is charting by an infinite number of dimensions. However, in a practical application, a requester with a specific list topic may only care about a finite representation of the idea space.

Suppose an idea space is represented by n vital dimensions $(d_1, d_2, ..., d_n)$ along with corresponding sets of values $(V_1, V_2, ..., V_n)$ where $d_{\mathrm{i}} \Rightarrow V_{\mathrm{i}}$, then a *cell* of the idea space is a combination of one value from each dimension $\{(v_1, v_2, v_3, ..., v_n) : v_1 \in V_1, v_2 \in V_2, ..., v_n \in V_n\}$. For example, suppose the list topic "ways to a brick" has an idea space of $n = 2$ dimensions: "i*s destructive*" $\Rightarrow$ {"*yes*", "*no*"} (V_1) and "*location*" $\Rightarrow$ {"i*ndoor*", ...} (V_2), then one of the cells will be {("*yes*", "i*ndoor*") : "*yes*" $\in V_1$, "i*ndoor*" $\in V_2$}. By definition, each cell represents a *distinct* idea. Thus the maximum number of distinct ideas is equal to the number of cells:

$$NumberOfDistinctIdeas \leq \prod_{\mathrm{i}=1}^{n} ||V_{\mathrm{i}}|| \tag{3.1}$$

, where $||V_{\mathrm{i}}||$ is the number of values for the dimension d_{i}.

An idea space is *completely covered* when each cell is covered by at least one idea, and *uniformly covered* if the number of ideas covering each cell is roughly the same.

3.3.2 Constraint

An *empty* cell of the idea space means one cell that is not covered by any idea. To cover it, we need to guide participants with *constraints*. It is straight-forward to convert an empty cell $\{(v_1, v_2, v_3, ..., v_n) : v_1 \in V_1, v_2 \in V_2, ..., v_n \in V_n\}$ into a batch of constraints ($C = (c_1, c_2, ..., c_n)$) by replacing v_i with a constraint c_i, where c_i is in the format of "v_i, *and not* $\forall v \in V_\text{i} \backslash v_\text{i}, --$ *w.r.t* d_i". Here, $\forall v \in V_\text{i} \backslash v_\text{i}$ means every value in the set V_i except v_i. For example, if v_i is a value "i*ndoor*" from dimension "*location*" $\Rightarrow$ {"i*ndoor*", ...}, then it will be converted to a constraint *"Indoor, and not outdoor, underground, or outerspace – w.r.t location"*. In particular, for a dimension with Boolean values (e.g. "i*s destructive*" $\Rightarrow$ {"*yes*", "*no*"}), its values will be converted into human-readable constraints like "i*s destructive*" and "i*s not destructive*".

3.4 Iterations of Generating Dimensions and Values

Our design went through four iterations before it reached to a point where workers were able to generate *Dimension/Values* that could comprehensively categorize any ideas of a list topic. The timeline of the iterations is shown in Figure 3.3. Within each iteration below, we summarize the communication strategies that were used to guide the task design as well as the preliminary findings after testing with workers.

In this section, we only demonstrate the first three iterations while describing the last iteration in the next section.

3.4.1 Iteration #1: Plain Requirement

We began our instruction design by directly asking workers to think of the *Dimension/-Values* that could be used to categorize the given list of ideas. Workers were given a list of ideas for a topic (e.g., *"ways to use a brick"*), the textual description of procedural informa-

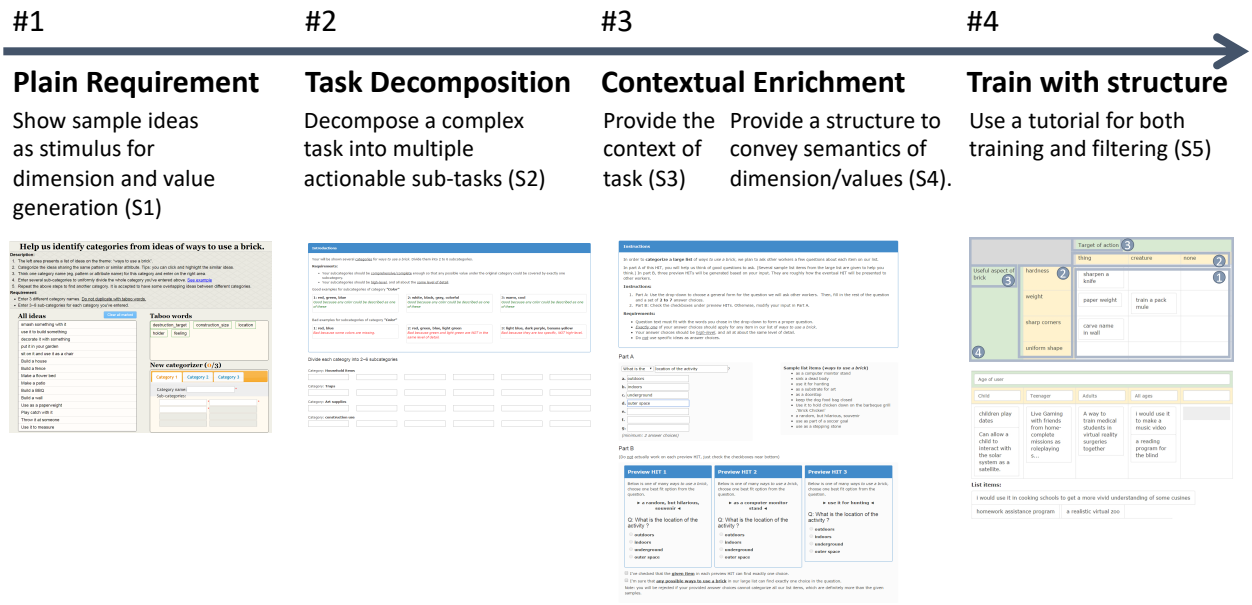

Figure 3.3. We went through four iterations of the design. In each iteration, we used one or two communication strategies to guide our optimization of the task design. This figure shows the progression of the design that finally reaches to a version (i.e., iteration #4) which effectively communicates the task to workers.

tion that need to perform to complete the task, as well as the requirements of the submission. This represents the communication strategy below.

Strategy: Show sample ideas as stimulus for dimension and value generation (S1). Showing sample ideas allows workers to focus on the generation of dimensions and values, without the need to think of ideas by themselves. Also, the sample ideas were often more diverse than those a worker could think of during a short period of time. Thus the diverse ideas were assumed to be more likely to inspire workers to generate more diverse dimensions and values. Cascade [135] also provided list items and asked workers to generate categories in order to form a taxonomy. However, one difference is that Cascade asked workers to generate one category for each list item, while we did not enforce a one-to-one mapping. Furthermore, one unique challenge is that workers had to provide dimensions and values that can be used to categorize ideas that are not in the list of sample ideas. This is an unusual task that requires relatively complex cognitive skills.

Table 3.1. Summary of statistics of Dimension from each Iteration. Note that a "Valid D/V" means the Dimension/Values pair that could be legitimately used to categorize an idea space. A dimension does not count as a valid "Valid D/V" if its corresponding values are invalid. The results were first coded by a researcher and then cross-checked by the other researcher.

Iteration	**Dimension**				
	Valid (%)	Invalid (%)	Irrelevant (%)	Valid D/V (%)	Total number
#1	9.3	65.1	25.6	7.0	43
#2	42.9	54.3	2.9	28.6	35
#3	52.4	39.3	8.3	44.0	84
#4	79.6	16.3	4.1	79.6	49

Note that we used the term "Category/Subcategory" rather than "Dimension/Values" in the task description. This was because we assumed that workers may feel more familiar with the former terms, given that categorization is a common task on crowdsourcing platform. However, since "Category/Subcategory" are overloaded terms and sometimes have conflicting meanings depending on the context, we also tried different terms in the later iterations, such as "Question/Choices", "Metric/Values" and "Dimension/Values".

Preliminary findings

We tested this set-up with 16 workers on Amazon Mechanical Turk (AMT), and received 43 dimensions and 156 values. We found several issues from the collected data. First, most dimensions were either invalid (28 out of 43) or irrelevant (11 our of 43) to the given topic (Table 3.1). Take dimensions of the topic "ways to use a brick" as an example, workers submitted dimensions such as "dangerous" (invalid) or "shape" (irrelevant), which cannot be used to categorize the ideas. By saying a dimension *invalid*, it means that it violates the criteria of a *valid* dimension defined in Section 2. Specifically, the dimension "dangerous" is considered as invalid because it can only be used to categorize a partial set of ideas in "ways to use a brick". The results were first coded by a researcher and then cross-checked by the other researchers.

Second, some dimensions are overly broad (e.g., "use", "usage"). Although these dimensions are reasonable, the granularity level of the corresponding value could either be very

broad, or extremely narrow that only cover one idea in the idea space. Besides the issues we found in dimensions, we also observed several problems from the values belonging to a dimension. For example, some values were wrong and did not belong to its dimension. Furthermore, even the values were correct, but they were not complete. As a result, only 7.0% of the D/V was valid. The results indicated that most workers did not thoroughly comprehend the task and submitted dimension and values that were unusable.

3.4.2 Iteration #2: Task Decomposition

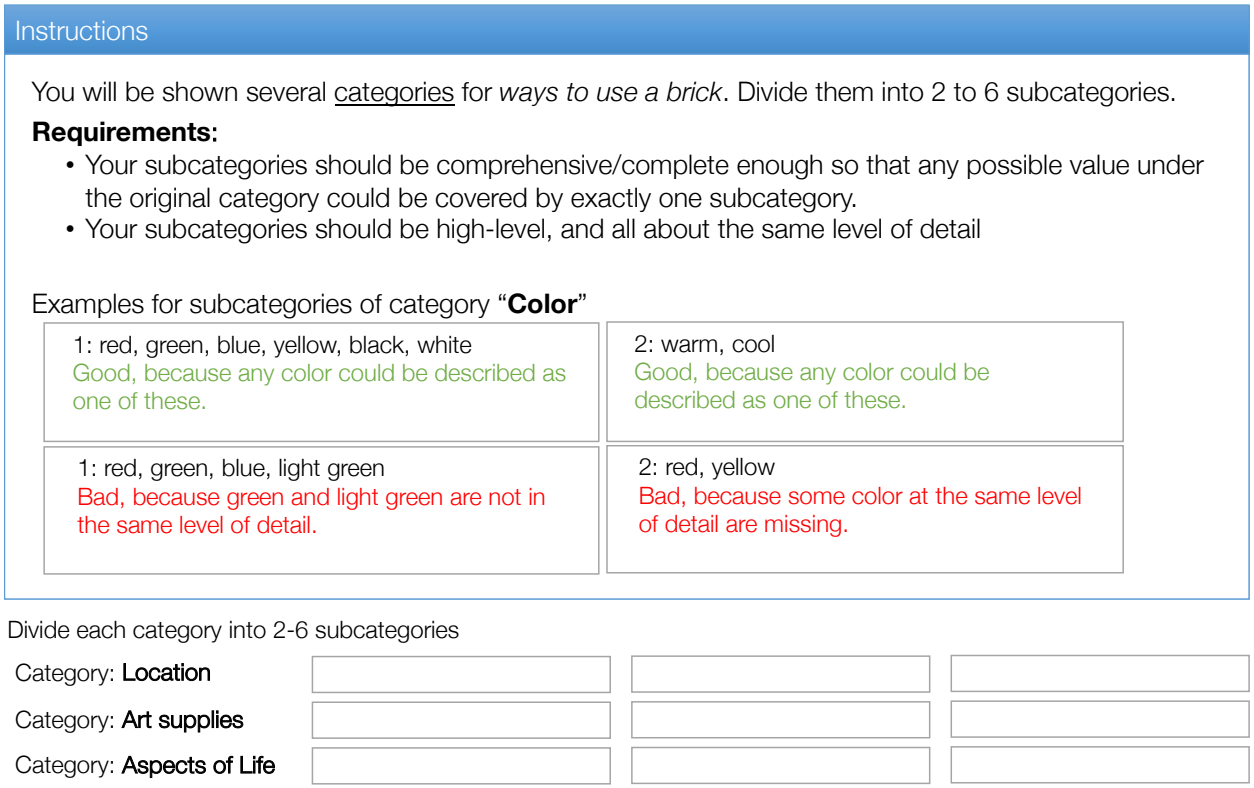

Figure 3.4. Interface of Iteration #2: Task Decomposition. Workers were asked to generate Values for a corresponding Dimension. Note that we use category/sub-category, not Dimension/Values, in the instructions to make workers more familiar with the term.

Our first attempt to improve workers' understanding of task was to reduce task complexity. In this iteration, we decomposed the original task into two sub-tasks. Specifically, instead of asking workers to provide dimensions and values at the same time, we simplified

the task and asked workers to either provide the dimension for a given list topic or provide values for a given dimension (Figure 3.4). In other words, workers only needed to focus on one job at one time. We adopted this strategy because we suspected that the original task might be too difficult for workers, given the failure in previous iteration.

Strategy: Decompose a complex task into multiple actionable sub-tasks (S2). Divide-and-conquer is a common algorithm that has been applied to divide a complex crowdsourcing task into several actionable units that could be performed by crowd workers [20], [136]. We did not consider this strategy at first because we thought the complexity of the original task is appropriate for crowd workers. However, after several unsuccessful attempts to optimize the instructions requirement in iteration #1, we suspect that the task might be too difficult to communicate with workers.

Preliminary findings

We tested this set-up with 10 workers on Amazon Mechanical Turk (AMT), and received 35 dimensions and 118 values. As Table 3.1 shows, we see a significant increase in valid dimensions (9.3% $\rightarrow$ 42.9%), and decrease in irrelevant dimension (25.6% $\rightarrow$ 2.9%). However, the "Valid D/V" is merely $28,6\%$, meaning that a large portion of the values were not aligned with the corresponding dimension, and vice versa. We found that it is very difficult for one worker to generate a dimension that another worker can build on to generate a set of values. This is because the second worker does not have the same understanding of context as the first worker. The second worker is often confused about what values to provide, given that the dimension quality is low. For example, if no values are provided, it is hard to understand the meaning of the dimension created by the first worker, such as "Aspects of Life", and thus hard to provide meaningful values.

Despite this possible confusion from workers, we did receive a better quality of D/V in this iteration, which shows the success of applying decomposition strategy (S2). On the other hand, the disconnection issue between the two sub-tasks (i.e., dimension generation and values generation) indicates that showing more contextual information of the delegated

task to workers might help improve workers' understanding of the task, and thus help them to provide results that meet requesters' implicit requirements.

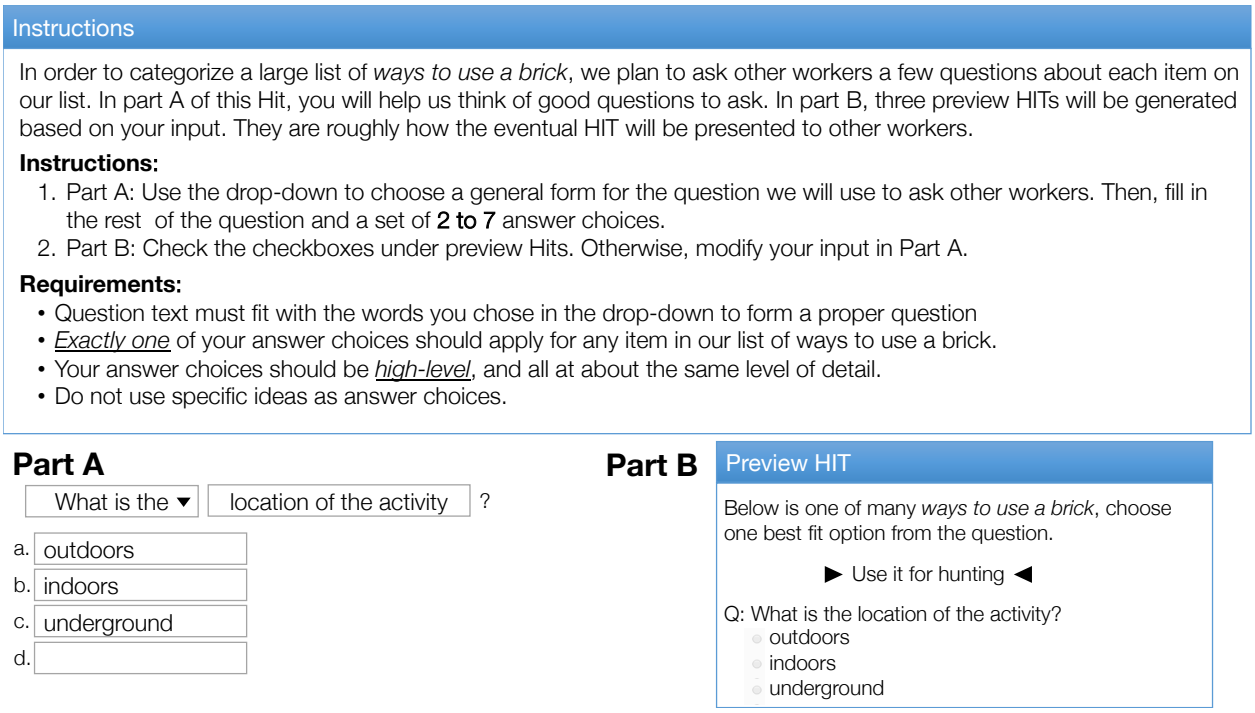

Figure 3.5. Interface of Iteration #3: Contextual Enrichment. We provide more contextual information of the task by letting workers know how their submission (Part A) would be used by other workers (Part B).

3.4.3 Iteration #3: Contextual Information Enrichment

Based on the findings from the previous iteration, a new prototype was designed and implemented to improve the quality of dimensions and values. As shown in Figure 3.5, workers were given a list of ideas as input and requested to think of a good question that can be used to ask **other workers** about each item in the list. Besides, workers were given three HITs that roughly shows how the eventual HIT will be presented to other workers. In other words, workers can *preview* how other workers would be asked to choose an answer for an idea with respect to the questions generated in this step. If they feel their question/choices do not fit with ideas, they are suggested to modify their question/choices. Two potential strategies were considered to guide the design of the new prototype.

Strategy: Provide the context of task (S3). This strategy is mainly inspired by our observation in previous iteration. When workers were finishing part of a complex task, they did not know how their contribution would be used by other workers. By giving the chance of previewing the context, they would be more willing to provide meaningful answers. We implemented three variations for S3, including 1) workers first provide a question and then preview the next step without doing it (as shown in the part B of Figure 3.5), 2) workers first provide a question and then actually do three previewed HITs of next step, and 3) workers first answer three previewed HITs that have been generated by others and then provide their own question/choices.

Strategy: Provide a structure to convey semantics of dimension/values (S4). We asked the workers to think of a question that is sensible for all the list items. We also provided a set of templates for the question, such as "With whom ___?", "What is the ___?" and "Is the [*list topic*] ___?" Workers needed to choose one of the template and fill the blank. We expected a worker can enter some questions like: "With whom can you use it?", "What is the location of the brick usage?" and "Is the way of using a brick expensive?" Furthermore, for each question, workers needed to enter two or more answer choices. For example, the question "What is the location of the brick usage?" may receive two answer choices: "indoor" and "outdoor".

The formed question is essentially a dimension, while the multiple choices of the question are essentially the values of the dimension. By representing the D/V relationship as a multiple-choice question, we aimed to leverage the intrinsic requirements for a good multiple-choice question: 1) each question should be sensible to the list items; 2) each answer choice should be relevant to the question; and 3) answer choices should be orthogonal (no overlapping), evenly-spaced, and as complete as possible. These requirements are well aligned with the criteria for a valid dimension and a valid set of values (as mentioned in the previous section).

Preliminary findings

We tested this set-up with 36 workers on Amazon Mechanical Turk (AMT), and received 84 dimensions and 273 values. Some good results were obtained, such as "What is the *time commitment*?" with four choices {"5-20 min", "21-45 min", "46-90 min", "95+ min"} and "What is the *physical effort*?" with three choices {"Little", "Moderate", "Heavy"}. Especially, when workers chose the template "Is the [*list topic*] ___?", they can often provide good *adjective phrases*, such as "Is the way of using a brick *expensive*?" with two choices {"Yes", "No"} and "Is the way of using a brick *fast*?" with two choices {"Yes", "No"}. The high accuracy of questions with adjective phases was largely because the answer choices were boolean values. Although the questions with adjective phrases had high accuracy, we did not encourage workers to solely generate questions in this type, because it is less informative than other types of questions and will cause too many dimensions. For example, a question "What is the *cost*?" with three values {"Low", "Medium", "High"} is equally informative as three questions: "Is the way of using a brick *cheap*?", "Is ... *medium-cost*?", and "Is ... *expensive*?".

As shown in Table 3.1, we found a 10% increase in valid dimension and 15.4% increase in "Valid D/V", compared to iteration #2. However, the dimensions (i.e., questions) and values (i.e., answer choices) still suffered from the issues of prior iterations. For example, some questions were irrelevant to the list topic (e.g., "What is the Capital of Canada?" for the topic "ways to use a brick") while some questions were too broad (e.g., "Is ... good?"). Additionally, some questions like "What is the most creative way?" were generated in which workers listed out several list items as answer choices. In this case, the answer choices of the question could not be used to categorize any items in the list.

We also found some issues different from prior iterations. Some questions like "What is the *best season for usage*?" with four choices {"Spring", "Summer", "Autumn", "Winter"} seemed to be a valid D/V. However, those values were not complete. Many list items of ways to use a brick do not have a clear "best" season for use. They can be used in full year. Similarly, a question was "What is the best time of day of usage?"

All the issues above were assumed to be resolvable by S3 but were not completely resolved. Showing previews of next step (Figure 3.5) did not help some workers to realize that their question/choices were irrelevant, too broad, or incomplete. The three variations of S3 did not have significant differences. We started to acknowledge that only some workers could pay enough attention to the requirements and fully understand the task, while the rest could understand partially or none.

3.5 BlueSky

We now discuss BlueSky (the system) and the algorithms that it uses to coordinate microtask workers to enumerate idea spaces.

3.5.1 Implementation

The backend of BlueSky is a Python web application using Web.py framework[1] (4.4k sloc). The database is using SQLite. The frontend of BlueSky is using JavaScript with Angular.js[2] library (6k sloc) and Bootstrap[3] CSS framework (version 3.3). The tasks are posted to Amazon Mechanical Turk[4] (AMT) where online workers can preview and accept the tasks. To communicate between BlueSky and AMT, a Python library called CrowdLib[5] is used.

3.5.2 Algorithm Steps

To begin, a requester enters a list topic as a noun phrase (e.g., "interview questions") accompanied by a brief description of the context (e.g., "Suppose you are the interviewer of an innovative company. . . .).

BlueSky's overall control flow consists of three HITs that are shown to participants. Here are the concrete instances of these HITs—*ideation* (Fig. 3.6), *synthesis* (Fig. 3.7), and *map*

[1]↑http://webpy.org/
[2]↑https://angularjs.org/
[3]↑https://getbootstrap.com/
[4]↑https://www.mturk.com/
[5]↑http://www.cs.umd.edu/hcil/crowdlib/

Algorithm 1 Enumerate idea space with crowd workers

Input: A list topic and a brief description
Output: A list of ideas that cover the idea space of the input topic

1. $S \leftarrow \emptyset$; // a set to store ideas
2. $D \leftarrow \emptyset$; // a set to store dimensions
3. $constraints \leftarrow \emptyset$;
4. i$sIdeaSpaceCovered \leftarrow false$;
5. **while** i$sIdeaSpaceCovered == false$ **do**
6. $S \leftarrow Ideate(constraints)$;
7. $S \leftarrow S \cup S$;
8. $D \leftarrow Synthesize(S)$;
9. $D \leftarrow D \cup D$;
10. $constraints \leftarrow Map(S, D)$;
11. **if** $constraints == \emptyset$ **then**
12. i$sIdeaSpaceCovered \leftarrow true$;
13. **end if**
14. **end while**

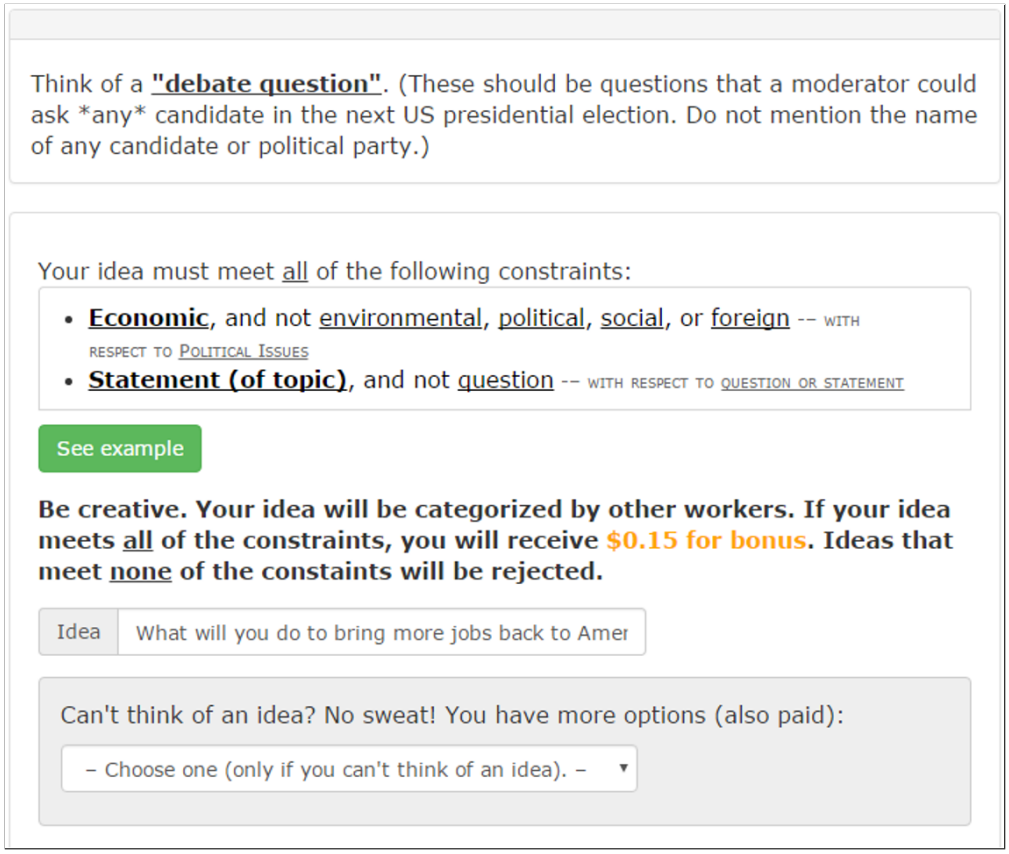

Figure 3.6. Ideation task interface with constraints

(Fig. 3.8). The system runs through a circular way, from ideation to synthesis, to map and then back to ideation.

BlueSky also includes a dashboard to assist requester to monitor the process and prune dimensions of no interest.

We first present an outline of the steps of Bluesky and then describe the steps in full detail.

1) Show list topic, description, and constraints (if any) to participants whom each **ideates** 1 list item based on the constraints (or 2 without constraints), or mark the constraints as N/A (not applicable).

2) Show list items to participants whom each **synthesizes** one dimension and 2-8 values that could be used to categorize the existing and future list items.

Figure 3.7. (Top) The tutorial for synthesis with four lessons. Area 1 teaches participants how to categorize ideas by given dimensions and values, while area 3 and 2 teach them how to generate dimensions and values, respectively. Area 4 teaches them how to combine all the skills in the previous lessons. (Bottom) The task interface for synthesis.

3) Show list items and dimensions to participants whom each **maps** the list items to their best fit value of each dimension.

4) Compute and generate constraints for empty cells. Repeat the algorithm (from step 1) with these constraints.

5) Prune (by requester) the dimensions of no interest in the dashboard during the whole process.

The algorithm continues until each cell in the idea space is either covered by list items or marked as N/A. Following are the implementation details for each step.

Step 1. Ideation with or without Constraints

This step has two phases, depending on whether participants are shown constraints or not.

Phase 1: The first several ideation HITs have no constraints as we have no inference on the idea space. Participants are asked to enter two ideas freely in each HIT.

Phase 2: After one round, participants will be guided by a batch of constraints that represents an empty cell in the idea space (Fig. 3.6). As mentioned in *Definitions*, the batch of constraints is denoted by $C = (c_1, c_2, ..., c_n)$. Participants are asked to enter one idea to meet every $c_i \in C$. For example, in Fig. 3.6, the HIT has a batch of two constraints ($C = (c_1, c_2)$) in which c_2: *"Statement (of topic), and not question – w.r.t question or statement"*. Note that c_2 is converted from a value "*statement (of topic)*" of dimension "*question or statement*" $\Rightarrow$ {"*statement (of topic)*", "*question*"}.

If participants have difficulty in generating ideas for the given constraints, they have two other options. One is to start over with another batch of constraints (e.g., $C = (c'_1, c'_2, ..., c'_n)$), namely, another empty cell. The other is to mark the current constraints as N/A if they are impossible to generate an idea, which in our case may be caused by incompatible combinations of dimensions or unclear values; meanwhile, participants need to write down the reason. If at least two participants mark the same batch of constraints as N/A, the corresponding cell in the idea space is marked as N/A and will not appear in the future HITs.

Based on the usability feedback of Phase 2, we added several optimizations. First, participants will start with a simpler batch of constraints. Specifically, they will be given a subset of two constraints from the batch C, e.g., $(c_i, c_j) \subset C$, as a stepping stone. After participants have done all the subsets, they will be given the full batch C. A subset contains two constraints because it represents a minimum combination of dimensions and can be used to test

compatibility. If a subset is compatible, the ideas generated for it will be used as examples for any batch C containing it (when participants click the button "Show example"). On the other hand, if the subset is incompatible, any batch C containing it will be automatically marked as N/A.

Second, to motivate participants to generate ideas covering all of the constraints in the batch, we provide them a bonus of $0.15 (US dollars) if their ideas are being categorized (in step 3) by the same values as the constraints. The bonus incents them to try hard while leaving us leeway to approve HITs where a worker makes a nominal effort but fails to meet all of the constraints.

Step 2. Synthesize Dimensions and Values

This step needs participants to enter one dimension and 2-8 values. Since the dimensions are fundamental for the idea space, we went through four iterations as mentioned in the prior section. The last iteration is to provide a tutorial with four lessons to explain the concepts of dimensions and values, as shown in the top of Fig. 3.7. The first lesson (marked by number 1) asks participants to categorize a list of "ways to use a brick" by given dimensions and values. The second lesson (marked by number 2) asks participants to enter values for given dimensions and already categorized list items. Likewise, the next lesson (marked by number 3) lets them enter dimensions for given values and already categorized list items. The last lesson (marked by number 4) wraps up the previous lessons: it first gives several list items (uncategorized) and requires participants to think of dimensions, values and then categorize those items.

We partition the necessary skills—categorization, generating dimensions, and generating values—into the first three lessons so that participants can build up skills gradually. Each lesson has the same list topic but different dimensions, values and list items, which makes the skills more generalizable. Once participants complete any of the first three lessons, their results will be compared with gold standard and prompted with explanation if incorrect. The results of the fourth lesson will be manually checked by a qualified participant who has passed the tutorial.

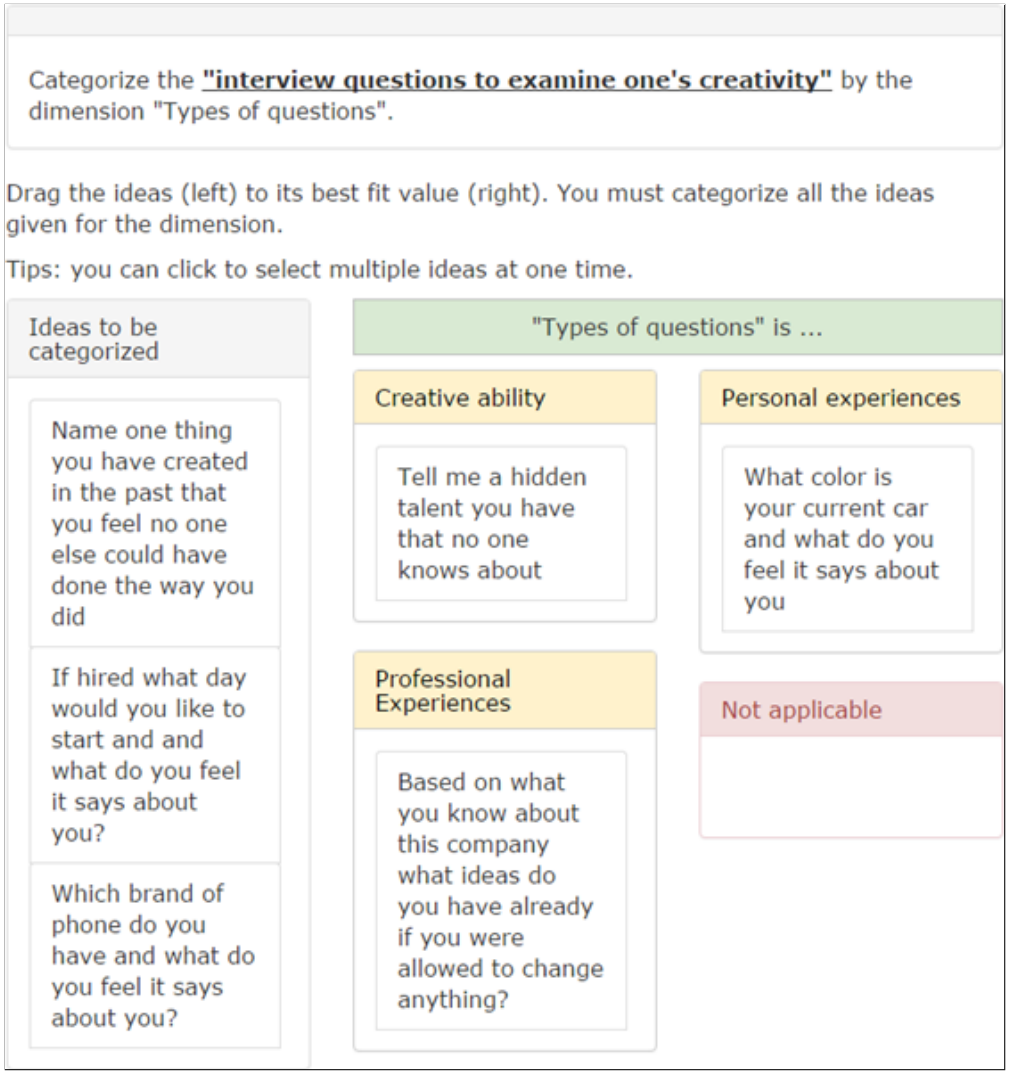

Figure 3.8. Mapping task interface

The task interface is similar to the fourth lesson of the tutorial, except that there is only one dimension (the bottom of Fig. 3.7). Only those who have passed the tutorial can take this HIT. Participants can see the tutorial they have done and thus have a strong connection between the tutorial and real task.

This step will be skipped when all of the important dimensions have been figured out. In case of pruning an important dimension by the requester, this HIT will restart after that.

Step 3. Map with Multi-judgment

This step needs participants to map the list items from step 1 to a set of values for each of the dimensions from step 2 (Fig. 3.8). It is essentially categorization.

Participants will see a *confirmed* dimension and several uncategorized list items (excluding ones that they entered in step 1) in each HIT. *Confirmed* means that the dimension is one of the vital dimensions (dependent on the requester) that could represent the idea space. If dimensions are provided by the requester, they are automatically confirmed. If they are provided by AMT workers, the requester may prune some nonsense dimensions or less interesting ones. In case of absence of the requester, dimensions provided by workers could be confirmed through multiple judgments: workers will see all of the dimensions and select the best one to categorize with; then the dimensions selected by three or more workers are confirmed. In our test, we found that the dimensions selected by three workers were usually selected by a fourth worker, so it is reasonable to ask three workers to perform the multiple judgment. At any point in time, the requester may come in and modify (or even prune) a confirmed dimension to fit their interests.

Similarly, a list item is categorized by a specific value if at least two workers categorize it with the same value. A list item can be categorized by only one value of a dimension, as the values are orthogonal to each other. Participants may also categorize a list item as N/A; if two or more workers categorize an item as N/A, it is marked as N/A for this dimension.

To reduce the number of mapping workers required, we use a simple optimization. For each dimension in the constraint at the ideation step, we count the ideation worker's assertion that the idea conforms the same as if a worker in the mapping step had evaluated it. However, since some contributed ideas do not truly conform to all of the values in the constraints, we still require one judgment from a mapping worker before we consider that cell in the idea space as filled.

Step 4. Idea Space Computation

After categorizing existing list items, BlueSky computes the status of the idea space. It stops running if all of the cells have been covered or marked as N/A, or posts more ideation HITs (step 1) for empty cells. This step is done automatically without participants.

Step 5. Dashboard for Requester

Through the dashboard, a requester can see the whole picture of the idea space, such as the covering status of each cell. Moreover, a requester can prune dimensions that are trivial or unlikely to generate useful ideas, and thus figure out the important ones.

3.5.3 Modeling Time and Cost

Here we estimate the time and cost to obtain a uniform sampling of the idea space with or without BlueSky.

First, suppose crowd workers generate ideas freely without BlueSky. From equation 3.1, we can calculate the total number of empty cells, for simplicity, denoted as N. A worker can generate one idea per HIT. Since ideation is not guided, the idea has a chance to cover any cell. The probability to cover the i-th cell is denoted as p_{i}, in which

$$\sum_{\mathrm{i}=1}^{N} p_{\mathrm{i}} = 1 \tag{3.2}$$

For j-th idea, the probability to cover i-th cell is denoted as p_{ij}, so that

$$\sum_{\mathrm{i}=1}^{N} p_{\mathrm{ij}} = 1 \tag{3.3}$$

Suppose workers need to at least generate M total ideas to completely cover all cells, then at the (M-1)-th idea, we must have one cell that has not been covered, say i-th cell

(the hardest to cover). In other words, the probability of i-th cell that has not been covered is

$$Pr(missing\ cell_{\mathrm{i}}) = \prod_{\mathrm{j}=1}^{M-1} (1 - p_{\mathrm{ij}}) \tag{3.4}$$

Suppose we want to estimate M when $Pr(missing\ cell_{\mathrm{i}})$ is below some threshold, say 0.01. In the simplest case, all p_{ij} are equal, thus $p_{\mathrm{ij}} = 1/N$.

$$\prod_{\mathrm{j}=1}^{M-1} (1 - 1/N) < 0.01 \tag{3.5}$$

or

$$(1 - 1/N)^{M-1} < 0.01 \tag{3.6}$$

so that

$$M > log_{1-1/N} 0.01 + 1 \tag{3.7}$$

For example, when N is 30, we have $M > 136$.

However, in real world, some cells are very easy to think of, while some cells are very hard. For such challenging cells, it is possible that p_{ij} could be consistently low than $1/N$ for all ideas (j=1...M). This means that the total number of ideas will be much larger. For example, suppose the last (hardest) cell has p_{ij} equal to $1/3N$ for all (j=1...M), then

$$M > log_{1-1/3N} 0.01 + 1 \tag{3.8}$$

When N is still 30, we have $M > 413$.

Second, for BlueSky case. Since workers are guided to think of ideas for each cell, the p_{ij} will be amplified by factor α_{j}, so that

$$Pr(missing\ cell_{\mathrm{i}}) = \prod_{\mathrm{j}=1}^{M-1} (1 - \alpha_{\mathrm{j}} \times p_{\mathrm{ij}}) \tag{3.9}$$

Note that the amplified factor α_{j} varies with the complexity of constraints given to workers. Compared to thinking for N cells in the unguided case, the guided case only need

to think idea for one cell. In such case, it is very likely that $\alpha_j \times p_{ij}$ is close to a constant, with $\alpha_j \times p_{ij} > 1/N$.

Since the time and cost is proportional to the total number of ideas M, we can see that the BlueSky case can significantly reduce the time and cost.

3.6 Main Experiments

To test the performance of BlueSky, we ran the algorithm on four list topics and analyzed the ideas it produced.

3.6.1 Recruitment of Participants

In step 1 and step 3, participants were AMT workers with minimal approval rate of 90%, while in step 2, participants were AMT workers who have passed the tutorial. As mentioned earlier, workers' tutorial results (i.e., the fourth lesson) will be judged by a qualified participant who has passed the tutorial. In this experiment, we asked two graduate students to take the tutorial and then recruited one who passed to judge workers' results. In principle, it is also plausible to let AMT workers be the judge once they have passed the tutorial.

3.6.2 List Topics

We picked four open-ended list topics that belong to different types. Two of them were related to tangible things, such as HoloLens (Microsoft augmented reality devices) and a gadget for finding keys. The others were intangible concepts like interview questions and debate questions. The list topics are summarized in Table 3.2.

These list topics represented three kinds of user goals in which a user: 1) needs lots of distinct ideas (not necessarily complete), such as mathematical test questions; 2) needs completeness for technical reasons and/or future-proofing, such as test cases for machine learning; or 3) needs inspiration to ultimately find the best idea, such as business names.

Table 3.2. Two types (T: tangible, IT: intangible) of list topics with their potential applications.

Abbreviation	List topic	Types	Application
useHoloLens	"ways to use HoloLens"	T	needs lots of distinct list items (not necessarily complete)
interview	"interview questions to examine one's creativity"	IT	needs completeness for technical reasons and/or future-proofing
debate	"debate questions for the US election"	IT	needs completeness for technical reasons and/or future-proofing
textAd	"text ad blurbs for a new gadget for finding lost keys to appear in Google Ads"	T	needs inspiration to ultimately find the best idea

3.6.3 Method

For each list topic, we conducted an experiment with two conditions: a *treatment* condition that gave the constraints generated by BlueSky and a *control* condition in which no constraints were given. The list items from the control were categorized by the same dimensions as the treatment condition. Participants were randomly assigned to one group or the other and were prevented from participating in both.

3.6.4 Results

In phase 1 of step 1, both the treatment and control posted the same number of HITs. However, in phase 2, the treatment HITs were twice as the control because if the constraints were given, workers needed to enter one list item instead of two. Those list items were categorized by the same dimensions. We stopped both the treatment and control groups when the idea space was completely covered or when the coverage remained the same for a long time (e.g., longer than two iterations). Therefore, we ended up with approximately the same number of list items for both conditions. For simplicity, we truncated the list items of two conditions to the same number. The results are shown in Table 3.3.

Table 3.3. Results for four list topics (for both BlueSky and control).

List topic	# of ideas	# of dimensions (confirmed/total)	# of total cells	Sample dimension&values
useHoloLens	308	4 / 16	128	"*Age of users*" ⇒ {"*child*", ...}
interview	106	3 / 13	36	"*Types of questions*" ⇒ {"*Creative ability*", ...}
debate	136	4 / 16	40	"*Foreign vs. domestic*" ⇒ {"*foreign policy*", ...}
textAd	134	5 / 10	32	"*Mentions keys?*" ⇒ {"*yes*", "*no*"}

useHoloLens produced 309 list items in the treatment condition by 96 workers and 308 items in the control condition by 40 workers. We truncated them into 308 items for simplicity. In both conditions, some list items were literally duplicate, such as "playing video games." Some were phrased differently but had the same meaning, such as "virtual museum tours" and "To give a tour of a museum to people remotely." Some were phrased differently with different meanings, such as "Explore places on Earth where humans can't survive." ($L1$) and "Simulate real world emergencies to respond to" ($L2$).

useHoloLens also produced 16 dimensions based on the list items by 6 workers who have passed the tutorial. In step 3, workers confirmed four dimensions by multi-judgment, such as "Age of user" with 4 values {"child", "teenager", "Adults" and "All ages"}. The other three dimensions had 2, 4, and 4 values, respectively. Thus the maximum number of cells in the idea space was 128; namely, there would be at most 128 distinct list items. In such case, even some items phrased differently with different meanings were treated as duplicate if they are categorized by the same value of each dimension. For example, $L1$ and $L2$ mentioned above were duplicate because they were categorized by the same value of each of four dimensions, such as "*Age of user*" ⇒ "*Adults*".

interview produced 106 list items in treatment and control conditions, such as "What would you do if it would start raining fish?", by 59 and 15 workers, respectively. Eight workers produced 13 dimensions and three dimensions were confirmed, such as "Types of questions"

with 3 values {"Creative ability", "Personal experiences", and "Professional experiences"}. The other two dimensions had 3 and 4 values, respectively, charting an idea space of 36 cells.

debate produced 136 list items in treatment and control conditions (after truncation), such as "Will you force all public washrooms to be inclusive to all genders, including transgendered persons?", by 79 and 18 workers, respectively. Seven workers generated 16 dimensions and four dimensions were confirmed, such as "foreign vs domestic issues" with 2 values {"foreign policy", "domestic"}. The other three had 2, 2, and 5 values, respectively, charting an idea space of 40 cells.

textAd produced 134 list items in treatment and control conditions, such as "How rich would you be if you had a dollar for every time you lost your keys?", by 58 and 28 workers, respectively. Three workers produced 10 dimensions in which five were confirmed, such as "Mentions keys?" with 2 values {"yes", "no"}. The other four dimensions all had 2 values, charting an idea space of 32 cells.

The size of the idea spaces is determined not by the topic, but by the chosen dimensions and values. Workers decide how many values to partition each dimension by. Requesters control which dimensions to accept. Together, they determine the number of cells into which the idea space is partitioned.

Time and Cost

Step 1 was paid $0.25 per HIT while step 3 was paid $0.30 per HIT. After the system stopped, the average cost per list item under the treatment condition was $0.56, $0.58, $0.53, and $0.55 for useHoloLens, interview, debate and textAd, respectively. Correspondingly, for the control, the average cost per list item was $0.44, $0.33, $0.31, and $0.39. The control group was slightly cheaper than the treatment because each control ideation HIT got twice ideas as the treatment with the same amount of payment.

Under the treatment condition, participants spent a combined total of 17 hours 9 minutes, 5 hours 31 minutes, 6 hours 49 minutes, and 7 hours 10 minutes for useHoloLens, interview, debate and textAd, respectively. Correspondingly, for the control, the combined work time was 7 hours 37 minutes, 2 hours 13 minutes, 2 hours 11 minutes, and 2 hours 38 minutes,

respectively. The control group was faster than the treatment group because workers in the treatment spent more time in ideation to meet the given constraints. Due to the same reason, there were more unique workers in the treatment (73 on average) than in the control (25 on average) where workers were able to add ideas quickly and continuously in multiple HITs.

The average hourly rate for the BlueSky condition was $10.46/hour. For the control, it was $18.07/hour.

3.6.5 Evaluation

The goal of BlueSky is to produce a list of items that completely and uniformly cover the entire idea space. To evaluate it, we ask and answer two key questions:

1. How comprehensively is the idea space covered?
2. How uniformly is the idea space covered?

Comprehensiveness

The systems stopped when the idea spaces of the treatment were covered by 111, 36, 36 and 27 cells for useHoloLens, interview, debate and textAd, respectively (Table 3.4). It indicates that the topic interview stopped after the idea space was completely covered, while others stopped after the idea space coverage remained the same for three iterations.

We stopped the control once the treatment stopped because it would take lots more time to cover some uncommon cells in the idea space. For instance, useHoloLens had a cell with the combination between "User category ⇒ government" and "Age of user ⇒ child". The treatment succeeded in covering it with creative list items (e.g., "*Children could use this to mimic a court room to learn about the judicial branch.*") while the control did not.

Correspondingly, in the end, the control groups had completely covered 59, 30, 16, and 20 cells, respectively. We perform a two-way ANOVA to compare the coverage (normalized by total cells) between treatment and control ($p = 0.06$), which is (barely) not statistically significant.

To measure how well a method covers the space while avoiding duplicates (same value for every dimension), the *efficiency* is calculated by the ratio of covered cells to the total list items (Table 3.4). Take useHoloLens as an example, the treatment had efficiency 111/308 = 36% while the control had 59/308 = 19%. The average efficiency of BlueSky (over four topics) was 29%, which was about 53% higher than that of the control (19%).

Table 3.4. Efficiency of BlueSky versus the control. Efficiency is defined as the number of distinct ideas ("# covered") divided by the number of idea entries in the ideation stage ("# of entries")

List topic	Group	Efficiency	# covered		# of entries
useHololens	BlueSky	**36%** =	111	÷	308
	control	19% =	59	÷	308
interview	BlueSky	**34%** =	36	÷	106
	control	28% =	30	÷	106
debate	BlueSky	**26%** =	36	÷	136
	control	12% =	16	÷	136
textAd	BlueSky	**20%** =	27	÷	134
	control	15% =	20	÷	134

Uniformity

The uniformity of an idea space can be illustrated by the distribution of list items covering each cell. Some cells in the idea space were more clustered than others and thus had more duplicate list items. The most clustered cells had 14, 8, 14 and 20 list items for useHoloLens, interview, debate, and textAd, respectively, and 53, 14, 30, and 30 for the four control groups. This indicates that the BlueSky reduced the heaviest duplication to about half of the control.

To illustrate how the BlueSky method enhances uniformity, the density of list items with respect to individual dimensions can be analyzed (Fig. 3.9). Note that uniformity on each dimension does not guarantee uniformity across all cells.

For the control, the dimension "question or statement" had a value "statement (of topic)" with the smallest density. It indicates that only a few list items were categorized by this value, and implies that this value might be a "blind spot" in the freeform ideation. On the

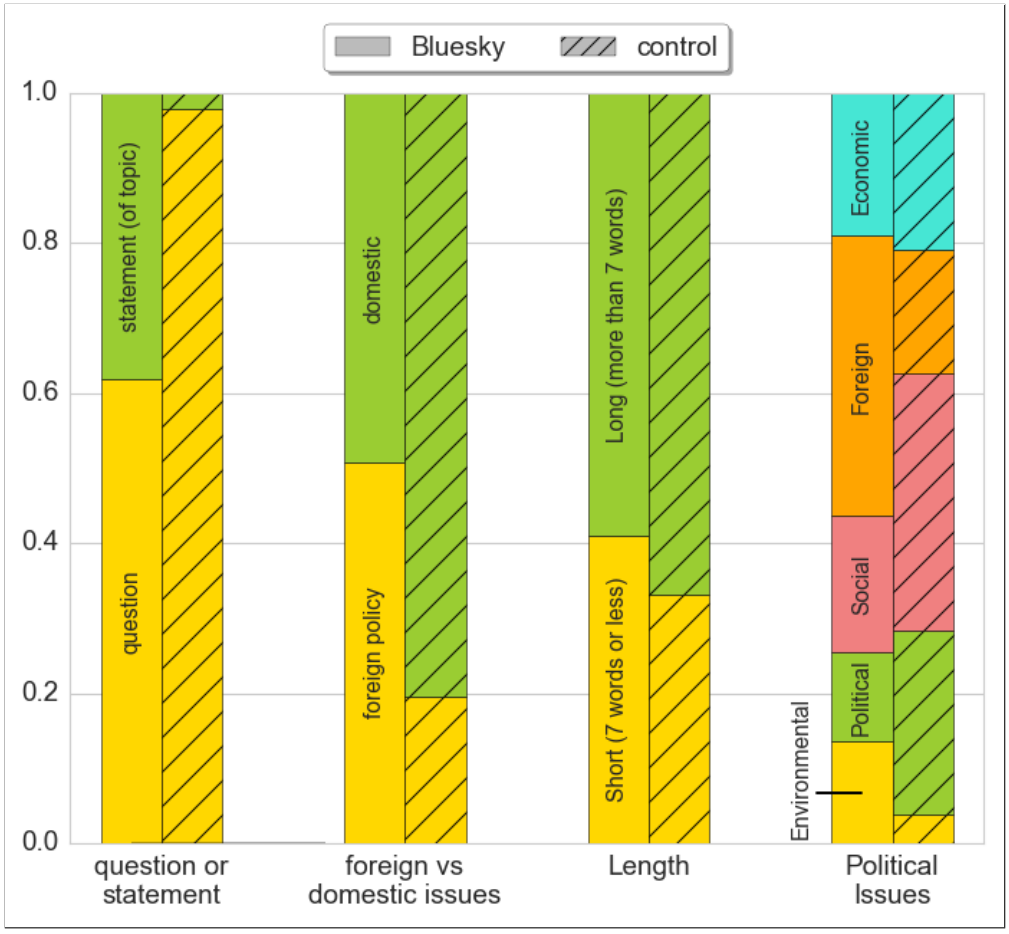

Figure 3.9. Value density of four dimensions on the list topic debate. The height of each column segment indicates the proportion of the total list items that were categorized with the value shown. If perfect uniformity could be achieved, then all segments within a given column would have equal height. BlueSky (left) is more uniform than the control (right).

other hand, this dimension had a value "question" with the largest density, which implies the participants' common thinking habits for the idea space. This situation is similar to other three list topics where BlueSky is more uniform than the control. By increasing uniformity, we could explore more uncommon (often valuable) list items as well as reduce the common (often boring) list items.

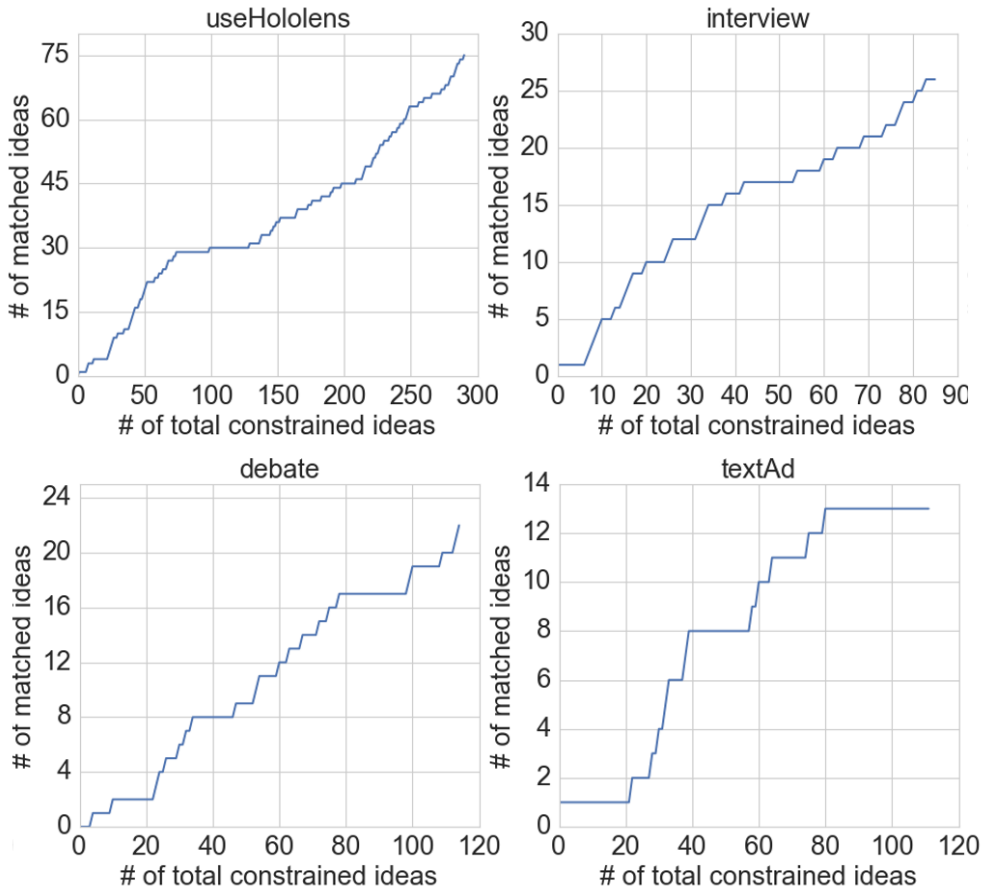

Figure 3.10. The number of matched list items over the number of total list items which have been given constraints for list topics under the treatment condition.

Match Rate of Step 1 and Step 3

In the treatment, a list item is generated w.r.t a batch of constraints and then categorized by one value of each dimension. Recall that each constraint (c_{i}) is converted from a value (v_{i}) of a dimension. If the categorization results (e.g., $(v_1, v_2, ..., v_n)$) match all the constraints C=$(c_1, c_2, ..., c_n)$ accordingly, this list item is called *matched* between step 1 and step 3. Workers could get a bonus of \$0.15 if their list item is matched. Figure 3.10 shows the number of matched list items over the number of total *constrained list items* (i.e., list items which have been given constraints). Interestingly, the number of matched ideas increased approximately linearly with the number of total constrained list items in most cases, except textAd. The

reason might be that the difficulty of each cell was similar, and workers generated a similar number of ideas to fully meet the constraints of each cell.

Note that the constrained list items were fewer than the total list items because some list items were generated with no constraints (phase 1 of step 1). Besides, they exceeded the number of total cells in an idea space because some constraints appeared more than once in the whole experiment. As mentioned above, each batch of constraints represents an empty cell and appears once in each iteration. If the empty cell is not filled in one iteration, the corresponding constraints will be posted again in the next iteration. The constraints will not appear when participants manage to generate a list item to meet the constraints (i.e., matched) or the empty cell is unintentionally filled by other list items.

Apparently, match rate is essential to cover an idea space efficiently. Ideally, if every list item is matched, the efficiency would be 100%. The actual match rate was about 20% on average. There are several possible reasons that could cause the relatively low match rate:

1. A list item was generated correctly (i.e., could meet all the constraints) but was categorized by different values in step 3. Since some dimensions have subjective values (e.g., "*Question answer length*" $\Rightarrow$ {"*short*", "*medium*", "*long*"} for the list topic interview), the participants in step 1 may have different opinions of "*short*" from those in step 3. This discrepancy could be mitigated, but not eliminated, by the multi-judgment in step 3.

2. A list item was generated incorrectly (i.e., could meet part or none of the constraints). It would either cover a non-empty cell (thus duplicate) or happen to cover an empty cell. It is not uncommon as some participants have difficulty thinking a list item to meet all the constraints. On average, 94% of list items could meet at least one constraint among the given batch, and 63.5% of list items could meet at least two constraints.

3.7 Discussion

BlueSky is a transformation from divergent to convergent process. The divergent process includes the freeform ideation with no constraints (phase 1 of step 1) and the dimension-

values synthesis (step 2). After all the dimensions and values are proposed, the convergent process leads the constrained ideation (phase 2 of step 1).

Most prior crowd-powered ideation systems aimed to maximize quality or creativity, or to find one great idea. In contrast, we aim to cover the space completely and uniformly. Thus, comprehensiveness and uniformity are the metrics that we used to measure the success of BlueSky.

The examples and constraints provided in step 1 may affect the creativity, especially when the constraints are difficult. There was no significant difference of match rate between workers who chose to see the examples or not.

The categorization task is also a common type of task on AMT. It is seen in other crowd workflows, such as Cascade [135]. Unlike Cascade, we only allow a list item to fit in one value, not multiple values because values are orthogonal (no overlap). If values are subjective or vague, it will depend on multi-judgment to find the best fit value for a list item. On average, 2.1 workers were needed to agree that a list item was categorized by one value (of a dimension), in which about 85% of categorizations were agreed by two workers and the rest were agreed by more than two workers.

In step 1, an option is to mark a batch of constraints as N/A, and less than 1% of workers chose it (then they would be asked to write down why the constraints were N/A). The low N/A rate is consistent with the fact that the entire space was (almost) completely covered and is also a sign of good dimensions. Similarly, in step 3, we allow workers to categorize ideas as N/A if they do not fit any of the values. Only about 1% of list items were categorized as N/A by more than two workers (multi-judgment). Most N/A list items were irrelevant to the list topic and/or made no sense (e.g., incomplete sentences).

3.8 Limitations and Future Work

The total running time of the system is long because it is sequential, not parallel. Participants have to wait until others finish the previous steps. If any participant spends an atypically long time (probably due to the confusion of instruction or holding the HIT but leaving), the whole system will be in idle. To speed up the system, a real-time ideation-mapping

workflow might help; namely, once a participant enters a new idea, another participant is notified to categorize it, and then new constraints will be given back. To that end, we might need to retain a group of workers.

Moreover, the more constraints in one HIT, the harder to meet them all. If participants choose to ignore some of the constraints, the system will slow down while the time and cost will increase since more HITs are needed. To increase match rate, we might filter/rank workers based on their performance. For example, we could invite the workers who have high match rates while block the ones with low rates.

Another challenge is efficient discovery of dimensions. One potential avenue would be to use machine learning to extract keywords or propose potential dimensions to inspire participants (including requesters). We could also maintain a pool of dimensions for each type of list topics. Participants could reuse and iterate existing dimensions or add new ones.

In this paper, we did not consider the hierarchy of dimensions. The values of a dimension might also be sub-dimensions. For example, "*location*" $\Rightarrow$ {"i*ndoor*", ...} could be further broken down into "i*ndoor*" $\Rightarrow$ {"*living room*", ...}. It might be interesting to allow zoom-in or zoom-out of the idea space.

3.9 Conclusion

This paper presents a crowd-powered algorithm that could mechanically enumerate a uniform and comprehensive sample of ideas via ideation-synthesis-mapping process. By testing with four list topics (useHoloLens, interview, debate and textAd) under the treatment (BlueSky method) and control conditions, we showed that the treatment could cover the entire space with 29% efficiency while the control could do with 19%. Moreover, the treatment covered the idea space more uniformly than the control, and reduced the heaviest duplication to about half of the control. The match rate between ideation and mapping steps was about 20%, suggesting that participants need to enter approximately five ideas to meet a given batch of constraints.

4. COSTORY: CREATIVE TASKS WITH FRIENDSOURCING

This chapter presents the second example of human-to-human task delegation, which is delegating creative tasks to friends or colleagues. This chapter has adapted, updated, and rewritten content from an unpublished paper coauthored with Alexander J. Quinn. All uses of "we", "our", and "us" in this chapter refer to coauthors.

When confronted with challenging interaction design problems, a common strategy is to enlist fellow designers and others for brief design assistance. They may bring complementary experiences and fresh perspectives on how to reconcile design constraints. However, unlike within-team collaborations or participatory design partners, these secondary contributors are not expected to be familiar with the anticipated use context—much less the rationale for the design choices up to that point. It is necessary to focus on just the part of the design problem that needs help. Building on prior work on supporting interaction design collaboration, we address the problem of informal collaboration with secondary contributors, with only limited interest and awareness of the design problem and draft solutions. This paper presents a framework for short-term *asymmetric collaboration* using a novel extension to interaction storyboards, in which a set of possible paths to the user goal is represented by a directed acyclic graph. This work was motivated by observations of research-oriented design projects, but the practice of soliciting short-term help from fellow designers is commonplace. Our evaluation found that the interaction storyboard hierarchies can improve the original design when the primary design team has already explored the problem sufficiently to identify well-defined subproblems about which to solicit assistance.

4.1 Motivation and Contributions

Interaction design requires considering who will use the products, how and where the products will be used, and what constraints will appear when interacting with the products. To better envision the potential interactions and constraints, a common practice is to consult fellow designers, colleagues, and others for brief design assistance. They may bring fresh perspectives and complementary experiences to help mitigate the design fixation. However, unlike within-team collaborations or participatory design partner, these *peripheral*

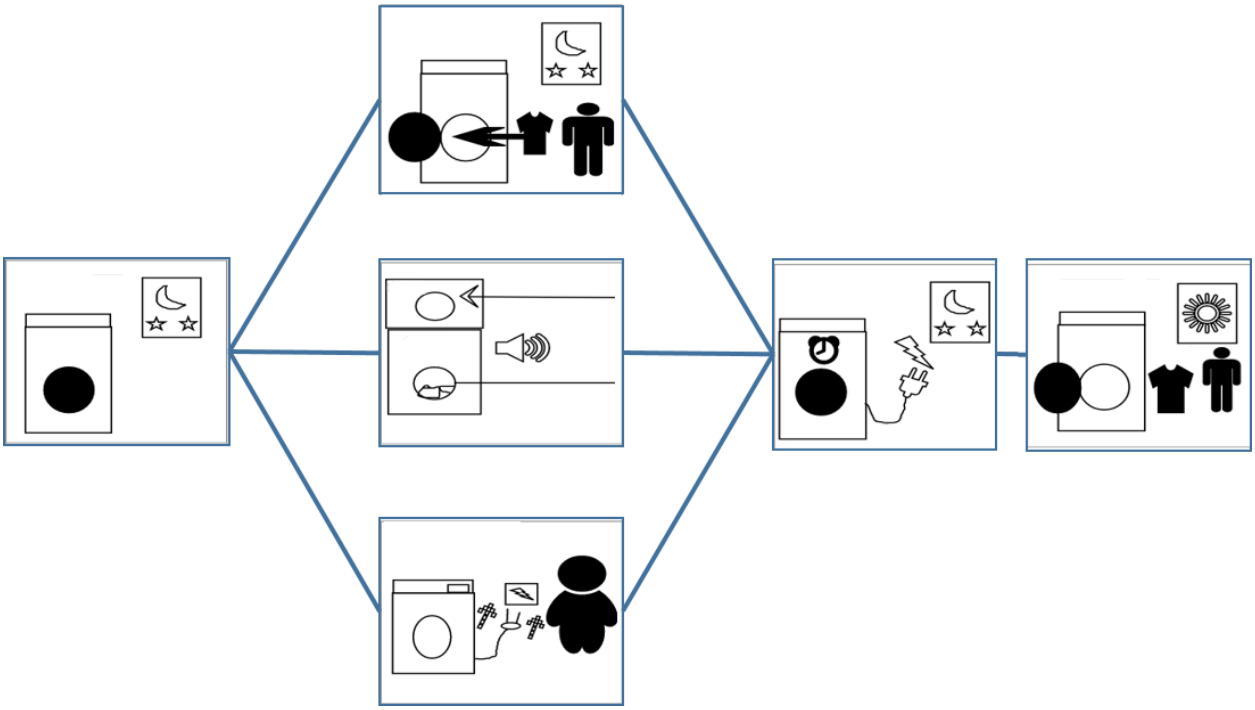

Figure 4.1. CoStory allows designers to request alternatives from contributors for key interactions. The panels above are part of a storyboard created by a participant in our user study. The key interaction is "how to setup the washer", which has three alternative designs.

contributors only have limited interest, responsibility, and awareness of the design problem and context (e.g., design choices and draft solutions).

We call this collaboration pattern as *asymmetric collaboration* in which peripheral contributors provide brief assistance to *primary designers* who are fully committed to the design goal and familiar with the design context. Asymmetric collaboration is a pattern that can be evidently found in scientific research [137]–[139], software engineering [140], and user-centered design [141], [142].

In the realm of interaction design, asymmetric collaboration is most likely to happen in ideation and prototyping stages of the four stages of interaction design (i.e., defining requirements, ideating alternatives, prototyping, and evaluating) [47]. These early stages welcome diverse ideas and fresh perspectives, and are suitable for peripheral contributors with limited involvement to provide beneficial assistance.

Due to the asymmetry of interest, responsibility, and awareness of the design problem, each session of asymmetric collaboration may be short. This short duration rises several challenges including how to focus on just the part of the design problem that needs help, how to define the necessary context to be shared with peripheral contributors, and how to bring peripheral contributors up to speed on the necessary context. More importantly, these challenges become worse when the design process keeps going and accumulates more context. Existing collaborative tools, however, lack the supports of facilitating this asymmetric collaboration, as they mainly focus on within-team collaborations (either globally or locally) [143], [144].

We present a process called *synthesize-request-ideate* to resolve the above challenges of asymmetric collaboration in interaction design. Specifically, primary designers first condense and *synthesize* the key parts of draft solutions, then *request* alternatives for the key part(s) from peripheral contributors, and finally peripheral designers *ideate* for alternative solutions. The synthesize step narrows down the context to share and identifies which part needs help. CoStory, a system we developed, implements this process. The key innovation is an *interaction storyboard hierarchy* in which multiple storyboards representing design alternatives become a directed acyclic graph while maintaining the flow-style appearance. We use storyboard as the medium because 1) storyboard provides a common language for 'read' and 'write' by both primary designers and peripheral contributors [55], and 2) the interactions with the products can be represented by sequential storyboard panels.

This research aims to boost the efficiency of asymmetric collaboration by a new storyboarding tool that supports context communication and problem-definition. Although storyboarding is chosen as the medium, the general principles are applicable to other design methods.

4.1.1 Contributions

The key contributions can be summarized as follows:

1. We summarize the pattern of asymmetric collaboration in general and its challenges with respect to interaction design. We propose a *Synthesize-Request-Ideate* process to resolve the challenges.

2. CoStory, a system we developed, implements this process to support efficient asymmetric collaboration via interaction storyboard hierarchies (Fig. 4.1).

3. Our evaluation found that the benefits of interaction storyboard hierarchies increase with the level of asymmetry between the design context mastered by primary designers and peripheral contributors.

4.2 The Asymmetric Collaboration Pattern

In this section, we summarize the pattern of *asymmetric collaboration* in general and its challenges with respect to interaction design. Moreover, we list the existing collaborative tools and discuss what they do not consider and what we can learn from them to resolve the challenges of asymmetric collaboration.

4.2.1 Asymmetric Collaboration

In the past experience of research projects, we consistently meet the need to solicit short-term help from fellows, such as statistic consultants, graduate students, and professors. Unlike within-team collaboration, these fellows only have limited interest and responsibility to our goals. Explaining the context and problems quickly is the key to make good use of the short-term collaboration session.

This collaboration pattern can be evidently found in many fields. For example, scientific research usually involves people who made specialized contributions (e.g., data, materials, and feedback), and may include them as co-authors or mention them in acknowledgements [137]–[139]. Similarly, software development, especially open source software developing (e.g., GitHub[1]), usually involves people with different level of investment, ranging from requesting new features, reporting bugs, writing code for a feature or bug, to becoming a core contributor

[1]↑github.com

[140]. Likewise, in design firms, designers often talk to other people, including prospective users and people outside of the team [141], [142].

We call this general collaboration pattern as *asymmetric collaboration* in which *peripheral contributors* with *limited investment* (e.g., time, energy, and effort) to the tasks/problems provide brief assistance to *primary designers* who are fully committed to the tasks/problems. The asymmetry of investment may be caused by limited motivation and commitment, and may result in asymmetric awareness of the context.

Asymmetric collaboration focuses on the "mental" aspect of investment, such as the time, energy, and effort devoted to the tasks, rather than the "physical" resources such as equipment and funding. This is narrower than a similar term "asymmetric resources" that describes the inequality of all resources during the collaboration between small firms and large partners [144], [145].

Moreover, asymmetric collaboration is not equal to "asymmetric roles" [146], "asymmetric disciplines" [147], "asymmetric information" [148] or "asymmetric expertise", because different roles, disciplines, information or expertise does not necessarily result in asymmetric investment, provided they have spent balanced amount of effort and time. On the other hand, asymmetric collaboration is likely to happen when people have different disciplines, roles, or expertise.

Asymmetric collaboration seems similar to *informal collaboration* [142], [149], [150], but has some key differences. Unlike formal and informal collaboration that is differentiated by whether the collaboration events are scheduled in advance or initiated impromptu, asymmetric collaboration emphasizes on the investment of participants. In other words, asymmetric collaboration can happen in both formal and informal collaboration. Moreover, informal collaboration usually happens synchronously in a shared workspace, while asymmetric collaboration can happen globally or locally, and synchronously or asynchronously.

4.2.2 Challenges w.r.t Interaction Design

Since asymmetric collaboration may happen in many different fields, their challenges may not be the same. Here we only discuss asymmetric collaboration in interaction design and

its challenges. In such case, requesters are the designers who lead the design process, so we call them *primary designers* as a more accurate name throughout the rest of paper. The challenges and possible solutions are summarized below.

Communication of Design Context

Since asymmetric collaboration is brief, it requires collaborators to quickly build a shared understanding of the design context. There are two potential challenges.

1. *Managing design alternatives efficiently.* It is often necessary and challenging to allow primary designers and peripheral contributors to *add, navigate, and compare* design alternatives efficiently. Design alternatives are created when designers are plotting potential interactions in different scenarios or trying out different options for a particular scenario. As more alternatives are created, managing alternatives becomes challenging. Several management methods may help, such as version history in StoryboardThat[2], revision paths in skWiki [151], juxtaposition in Calico [152], directed acyclic graph in [153], and hyperbolic tree visualization in IdeaVis [143].

2. *Specifying the problem that needs help and the context that needs sharing.* If a design has many alternatives, it is overwhelming for peripheral contributors to digest in a short time. Therefore, it is often necessary and challenging to clearly specify the part(s) that needs help and the context that is necessary to be shared. Several decomposition methods may help, such as morphological chart [154], functional decomposition [155], and cascading design [156], which can break a complex design into smaller ones. In particular, by adding hierarchies in cascading design [156], designers and contributors can navigate freely between different levels of detail and simplify the design process.

Motivation

Peripheral contributors may need motivation to participate in an asymmetric collaboration session. Some study suggests that participants are unlikely to collaborate effectively unless they see the value in working together over an issue [157]. In other words, some

[2]↑www.storyboardthat.com

Table 4.1. Existing computer-supported collaborative tools with respect to the method of communication, method of managing alternatives, method of decomposition, and the location to be used.

Tool	Method	Managing alternatives	Decomposition	Location
TEAM STORM [158]	Sketching	Juxtapose	No	Co-located
Tele-Board [144]	Sketching	Juxtapose	No	Distributed
IdeaVis [143]	Sketching	Hyperbolic tree	Hierarchy	Co-located
Calico [152]	Sketching	Juxtapose	Hierarchy	Both
skWiki [151]	Sketching	Revision paths	No	Both
StoryboardThat	Storyboard	History	No	Both
SVSb [159]	Storyboard	No	No	Co-located
COMuICSer [51]	Storyboard	No	Hierarchy	Co-located
Coeno [160]	Sketching	Juxtapose	No	Co-located

cognitive motivation, such as self-efficacy and/or confidence of making useful contribution, needs to be taken into account.

4.2.3 Existing Collaborative Tools

In Table 4.1, we summarize several collaborative tools with respect to the challenges of supporting asymmetric collaboration mentioned above. As mentioned in Related Work, storyboarding is useful in authoring and communicating design context. Therefore, we narrow the scope of tools to storyboarding tools. Since sketching tools have the potential to draw storyboards, we slightly broaden the scope and include them as well for a more comprehensive comparison.

Table 1 shows that those tools have used various methods to manage alternatives. Specifically, version history and juxtaposition typically represent each alternative as a complete version of storyboard, which means that the same storyboard panels may be duplicated into different alternatives (e.g., Calico [152]). On the other hand, a graph-based approach [153] saves alternatives as branches, which means that the same storyboard panel is a single node of the graph that is shared by related alternatives, instead of being duplicated. Such graph is typically a directed acyclic graph, in which each path in the graph represents a storyboard.

As the number of alternatives increase, the graph-based approach may be more efficient in navigation and comparison than version history and juxtaposition due to its compactness. For example, since alternatives are represented as branches of a graph, people can quickly understand how many branches have been created at which part of a graph as well as compare alternatives by comparing branches [153].

Table 1 also shows that a few tools have used hierarchy to manage the storyboards or sketches, such as IdeaVis, Calico, and COMuICSer. For example, Calico used a three-level hierarchy, ranging from the concrete level of canvas that supports sketching, to the cluster level that groups canvases, and to the highly abstract level that groups the clusters. Moreover, IdeaVis used a hierarchy to hold a group of variants for a single sketch. Lastly, COMuICSer used a hierarchy to maintain the connections between every artifact and the part of the storyboards it refers to, in which the storyboard itself had no hierarchy.

While these tools used hierarchy in some aspect, they did not support hierarchy at the level of design alternatives. To be more specific, Calico represented the alternatives at the canvas level in which all alternatives were juxtaposed without further hierarchy. Likewise, COMuICSer had no hierarchy in storyboard itself, as mentioned above. IdeaVis seems to support hierarchy at the level of design alternatives, but it is a hyperbolic tree structure. Such tree structure is suitable for alternatives of sketches, but not suitable for storyboards because the panels of storyboards are closely related to the previous and next panels.

To leverage the benefits of graph-based approach in managing alternatives and to fill the gap of applying hierarchy to decompose design alternatives, we propose a different visualization called *interaction storyboard hierarchy*, which will be described later.

4.3 CoStory

CoStory was designed primarily to boost the asymmetric collaboration in the early stage of interaction design, i.e., design alternatives and low-fidelity prototyping. In particular, storyboarding is chosen as the authoring and communication medium.

This section describes our process to facilitate the asymmetric collaboration and the implementation of this process as a hierarchical storyboarding tool.

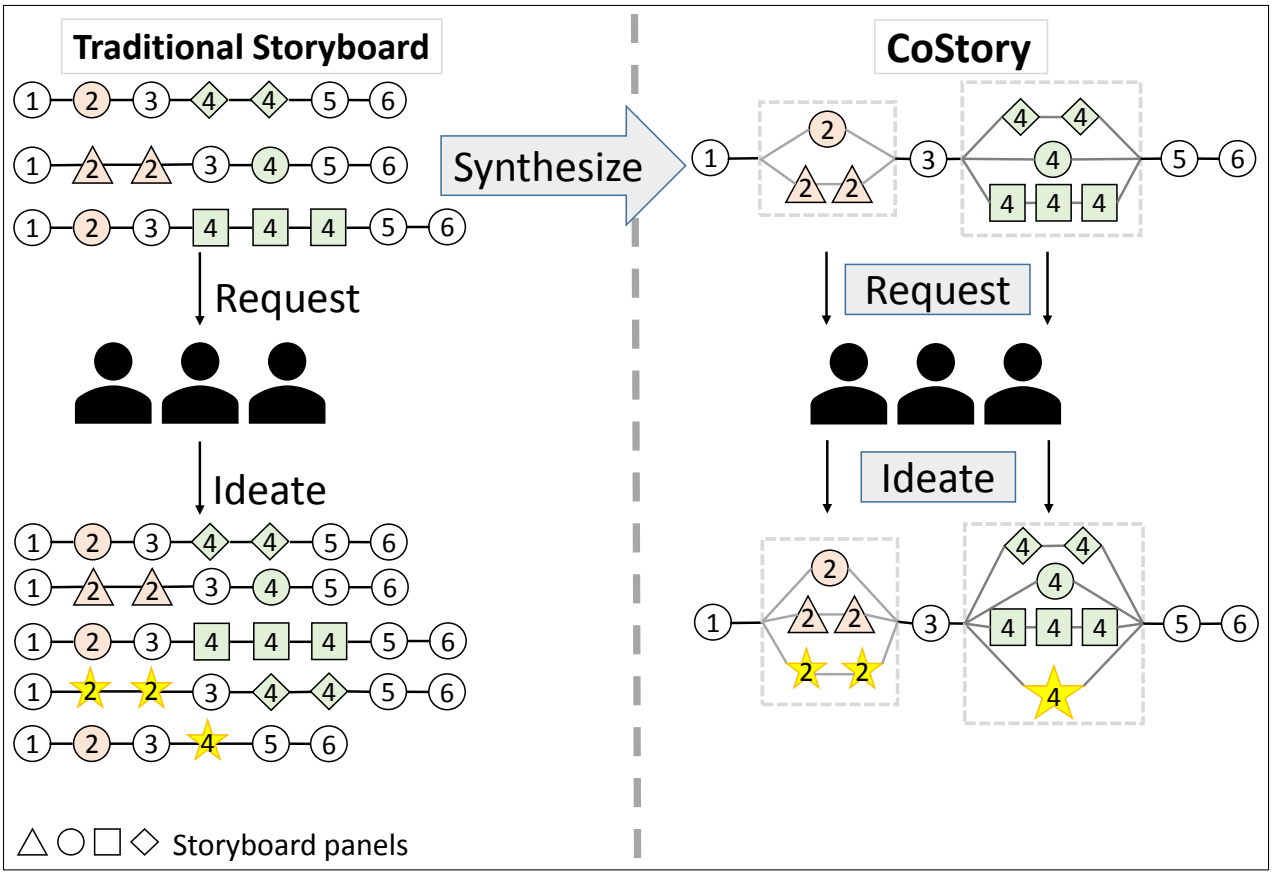

Figure 4.2. Synthesize-Request-Ideate process. Unlike collaboration with traditional storyboards, CoStory condenses the context first before sending it to peripheral contributors.

4.3.1 Synthesize-Request-Ideate Process

From the graph-based representation of storyboards mentioned earlier [153], we can see that some parts of the graph have more branches than other parts (as single nodes). Similarly, in interaction design, some parts of the interactions may be more critical than other parts, which deserves exploration of more diverse alternatives. We call those parts as *key interactions*, which may depict the interactions between prospective users and the core functions of products.

To bring more fresh perspectives for the key interactions, we propose a process called *Synthesize-Request-Ideate*, as shown in Fig. 4.2. Specifically, primary designers need to *synthesize* the design alternatives by merging them as a graph and further decomposing

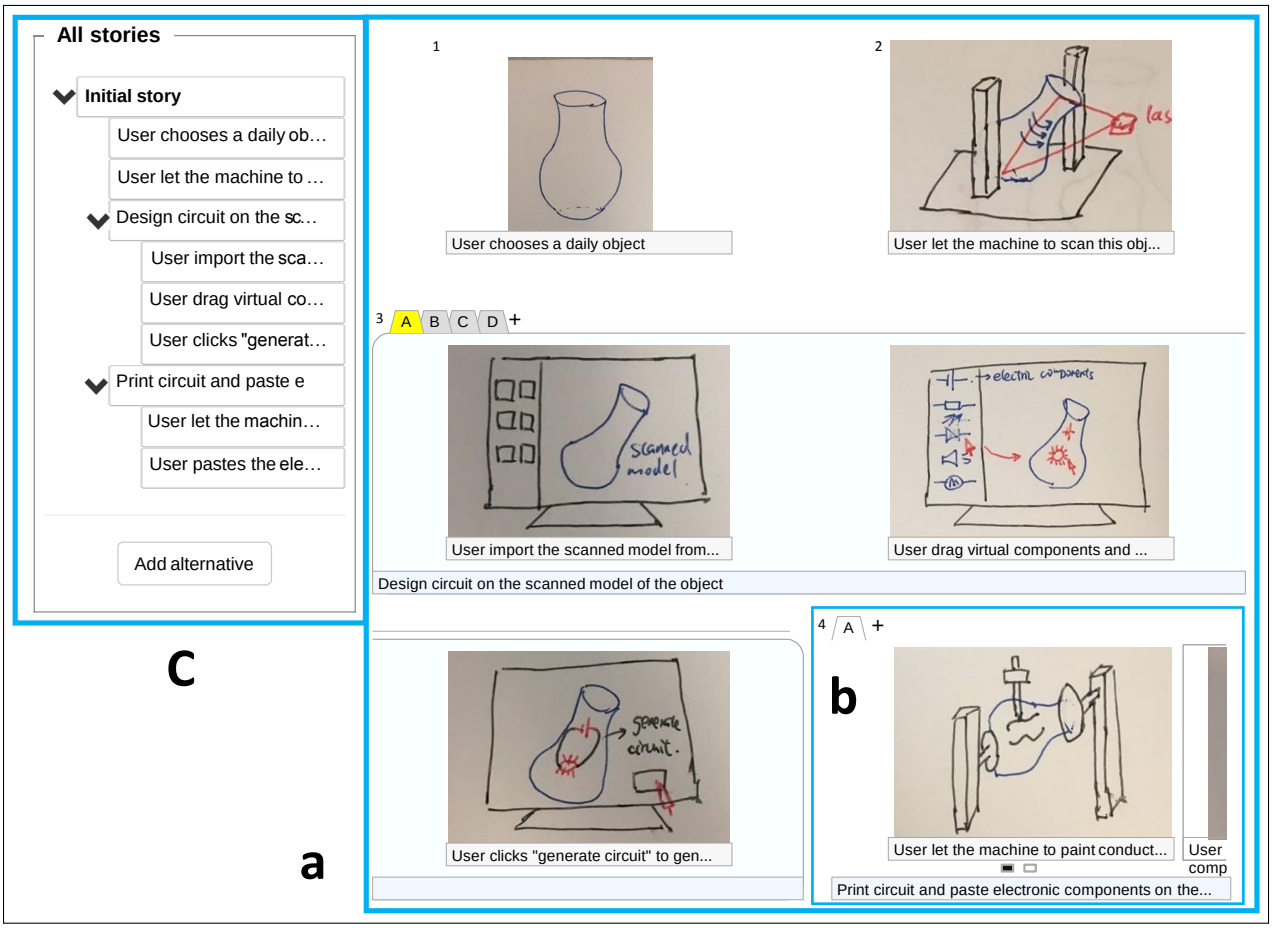

Figure 4.3. Primary designers' interface for the design task "Construct circuit on daily object". a) Main workspace. b) A cluster. c) Outliner. The storyboard and alternatives were created by participants in the user study.

the whole graph as several *clusters*. Each cluster represents one key interaction that has one or more alternatives, and each alternative has one or more storyboard panels. This step can leverage the benefits of the graph-based method of managing alternatives, and the hierarchy-based method of decomposition, as mentioned earlier.

Then primary designers need to *request* assistance from peripheral contributors by providing a well-defined problem description [161] for a particular cluster (i.e., key interaction). Finally, peripheral designers can *ideate* new alternatives for the requested cluster, based on the given synthesized context and problem description. The new alternatives will be added to the cluster automatically. In the next round, primary designers may further *synthesize* the design alternatives within each cluster, and repeat the steps of *request* and *ideate*.

Listing 4.1Algorithm for requesting designs from friendsourcing (pseudocode)

```
def main():
    initial_design = designer.draw()
    for each friend in designer.friend_list:
        scenario = request_scenario(initial_design, friend)
        clusters = divide_into_cluster(scenario)
        for each cluster in clusters:
            if cluster needs alternatives:
                request_alternative(cluster, friend)

def request_scenario(design, actor):
    if actor can provide a scenario for design:
        scenario = actor.draw()
        return scenario
    for each friend in actor.friend_list:
        # recursively subdelegate to friends of friends
        scenario = request_scenario(design, friend)
        yield scenario

# Similarly for request_alternative()
```

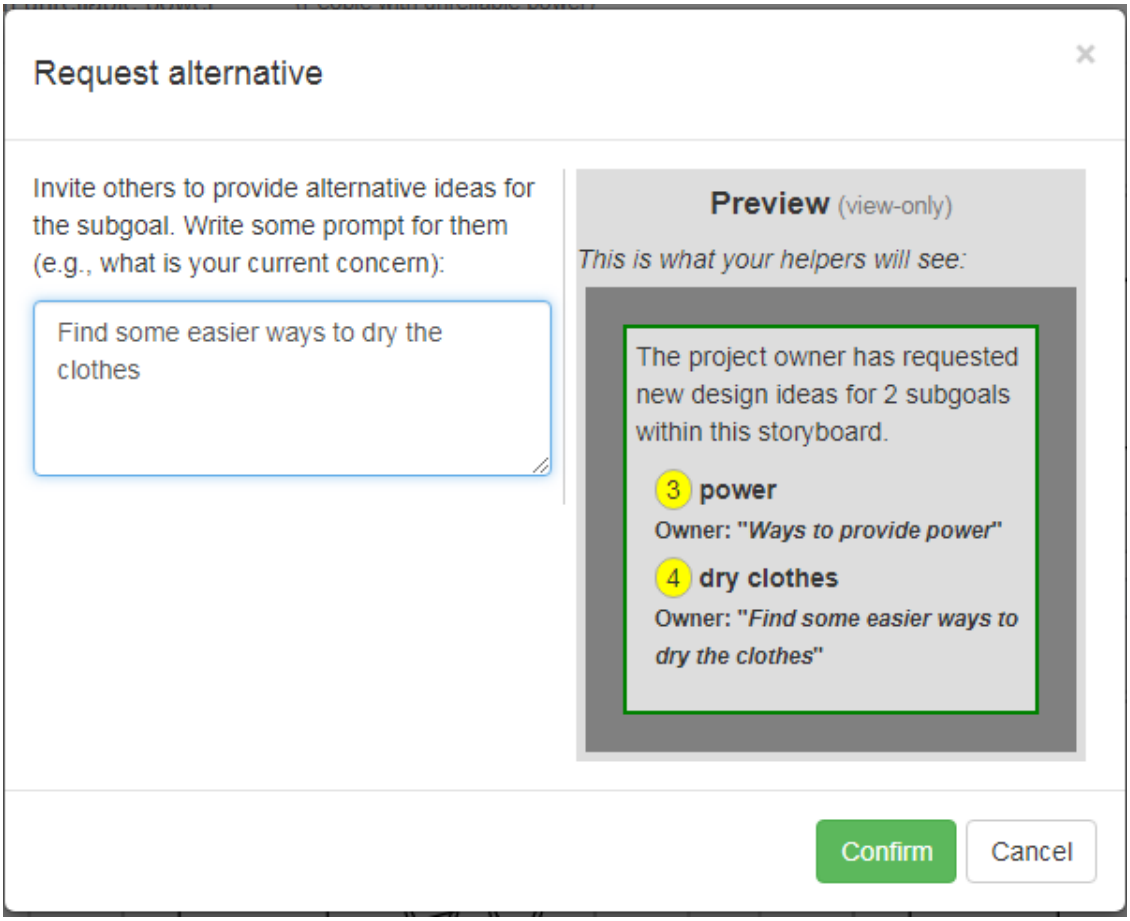

Figure 4.4. Primary designers' interface to request alternative for a cluster, write prompt (left) for peripheral contributors, and preview (right) what peripheral contributors will see later.

4.3.2 Implementation

CoStory implements the process above as a web-based application. The server side is using Node.js[3] (version 8+) and Express.js (version 4), with 1.5k sloc. The database is using MongoDB[4] (version 3.6). The client side is built on D3.js[5] (version 3+), with 5.7k sloc. We use Socket.io[6] to synchronize the updates between designers and contributors in real time.

Figure 4.3a shows the main *workspace* for primary designers.

[3]↑https://nodejs.org/en/
[4]↑https://www.mongodb.com/
[5]↑https://d3js.org/
[6]↑https://socket.io/

Create Panel

Each storyboard panel has a drawing and a caption. The drawing tool is adapted from an open-source tool[7]. This tool allows users to draw shapes from scratch, choose built-in shapes (such as icons for people, daily objects, and animals), and upload an external image. Besides, we allow users to draw with pen and paper, and then use phone camera to take a photo and upload it.

To accommodate the target users of CoStory, we did not implement it on a digital tablet or whiteboard that supports drawing with digital pen. This is because the target users may have different level of design expertise and may have no access to the digital tablet/pen at the moment of the collaboration. For example, primary designers could be professional designers or amateur designers, and peripheral contributors could come from different disciplines and may have plenty of or limited design expertise.

Create Clusters (Synthesize)

Primary designers can create a *cluster* to hold panels that are belonging to the same key interaction, as shown in Figure 4.3b. A cluster is a container that can hold one or more alternatives, representing as *tabs* on top of each cluster. One alternative (i.e., tab) can contain one or more panels, and even nested clusters. In such case, a storyboard can be represented as a hierarchy of clusters. Moreover, such cluster-augmented storyboard is essentially a directed acyclic graph but maintains a flow-style appearance. The flow-style appearance is chosen instead of a node-link graph due to several reasons. First, the former is more space efficient than the latter. Second, it is a common way to present storyboards so that designers have no difficulty understanding this conceptual model. Third, it inherits time-passing feature of storyboards in that the flow can represent the sequences without explicit links between panels.

A cluster also has a caption that provides a brief description of the key interaction. Consequently, it adds a level of abstraction that helps reducing the cognitive load.

[7]↑https://github.com/duopixel/Method-Draw

Unlike the cluster view used in Calico [152], our cluster is used at the panel level, while Calico used cluster at a higher level in which it contains different storyboards.

Request Assistance (Request)

Since each cluster represents a key interaction, designers can directly request alternatives for a specific interaction from peripheral contributors. Primary designers can write *prompts* for peripheral contributors in a pop-up window, as shown in Fig. 4.4. Prompts may explain what primary designers are not satisfied with the current design. To help primary designers establish an empathy with peripheral contributors when they face the prompts, designers are given a preview showing what peripheral contributors will see, as shown in the right of Fig. 4.4.

Ideate Alternatives by Peripheral Contributors (Ideate)

Peripheral contributors work asynchronously for the requested cluster. They have read-only access to the existing storyboard as well as the prompts. However, they have write access to the new alternatives created by themselves, as shown in Fig. 4.5. This will prevent contributors making trivial edits on the existing panels, such as changing fonts or colors. Also, by accessing all the existing alternatives of a cluster, it may reduce the chance of adding duplicate alternatives.

Once they finish drawing and click "Submit" button, this new alternative will be synchronized to primary designers immediately. Specifically, in primary designers' workspace, a new tab will appear automatically on the top of the corresponding cluster.

Compare Alternatives

By using tabs on top of a cluster, one can easily navigate and compare alternatives. Besides, they can view an alternative of a cluster in either *collapsed* or *expanded* mode.

Collapsed mode makes the space more compact and helps to reduce the cognitive load. Figure 4.3b shows an alternative in collapsed mode, in which only one panel is shown at the center, while others are hidden (partially or completely). On the other hand, expanded mode

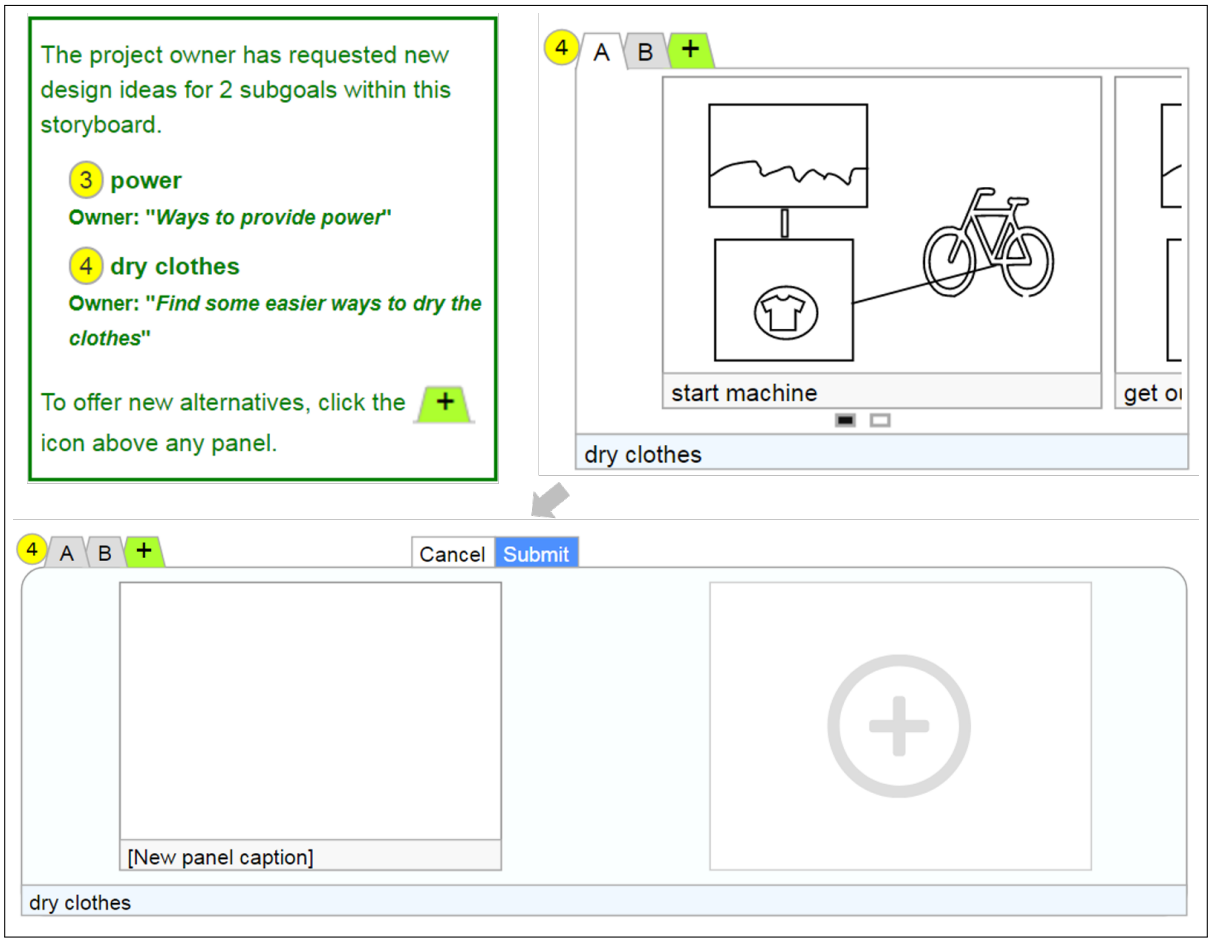

Figure 4.5. peripheral contributors' interface, including prompts from primary designers (top-left), and cluster with highlighted number and plus sign (top-right). The inner workspace (bottom) for adding new alternative is the same for both primary designers and contributors.

shows all panels of an alternative at the same time. Since an alternative may contain many panels, those panels may span across multiple rows after expansion, due to the flow-style visualization. To differentiate the expanded panels of an alternative from other panels, the panels will be wrapped by a rounded rectangle with background color, as shown in Fig. 4.3a.

Outliner

Outliner shows a higher level abstraction of the whole storyboard by showing the captions of panels and clusters in a hierarchical way, as shown in the Fig. 4.3c. It supports drag-and-

drop to reorder storyboard panels or clusters. It also permits users to nest a panel or cluster inside another cluster.

Users (either primary designers or peripheral contributors) may come up an idea that is completely different from the existing storyboard. They can click on "Add alternative" at the bottom of Outliner to create a new story. In such case, the new story will not share the same clusters as the previous one(s). This allows authoring, navigating, and comparing alternatives at the level of story, which is higher than the level of cluster.

4.3.3 Modeling Time and Results

Here we estimate the time for which primary designers have to wait after requesting alternatives from friends or colleagues. Suppose i-th helper receives the request, the probability of returning an alternative is p_{i}. Suppose the alternative has a chance of r_{i} to meet the intent of primary designers. Then, suppose M helpers are requested, the total meaningful designs is:

$$results = \sum_{\mathrm{i}=1}^{M} p_{\mathrm{i}} \times r_{\mathrm{i}} \tag{4.1}$$

The expected time to receive the responses from helpers is

$$E(Time) = \sum_{\mathrm{i}=1}^{M} (p_{\mathrm{i}} \times t_{\mathrm{i}} + (1 - p_{\mathrm{i}}) \times Timeout) \tag{4.2}$$

, in which t_{i} is the time spent by a helper who actually returns an alternative, while $Timeout$ is predefined maximum time to wait.

4.4 User Study

To evaluate whether the system can boot the efficiency of asymmetric collaboration, we hypothesized that cluster-augmented storyboards would (a) enable more efficient collaboration with peripheral contributors not directly responsible and (b) cause peripheral contributors to feel more confident about the value of their contributions, compared to unstructured collaboration. Specifically, we hold the following hypothetical benefits of the system and conduct a user study to test these hypotheses.

- H1: CoStory helps contributors understand the context better.
- H2: CoStory enhances contributors' confidence on their alternative.
- H3: CoStory helps contributors meet the intent of designers better.

4.4.1 Experimental Condition

To isolate the effects of the cluster-based design collaboration, we compared CoStory against a version of the same software with the clusters disabled. This is roughly comparable to many online storyboard authoring tools.

CoStory was tested in two different conditions:

- **CoStory (treatment):** The participants used a cluster-based storyboard to request alternatives.
- **Standard storyboard collaboration (control):** The participants used traditional panel-only storyboards for authoring and communication. This was achieved by curbing all the features related to clusters from CoStory.

4.4.2 Participants

We recruited 8 students (4 males, 4 females) across the campus, who have their own design problems that they have already put some thoughts in and need assistance from others. Participants were aged between 18 and 32.

4.4.3 Method

The eight participants were split into treatment group (T) and control group (C) based on their arrival order. The first four participants (P1-P4) were in group T while the rest (P5-P4) were in group C. Participants in both groups would follow the same steps: 1) reading tutorial customized for T or C in 20 minutes, 2) creating a storyboard for a design goal of their choice in 30 minutes, 3) contributing to three others' storyboards in Latin-square order

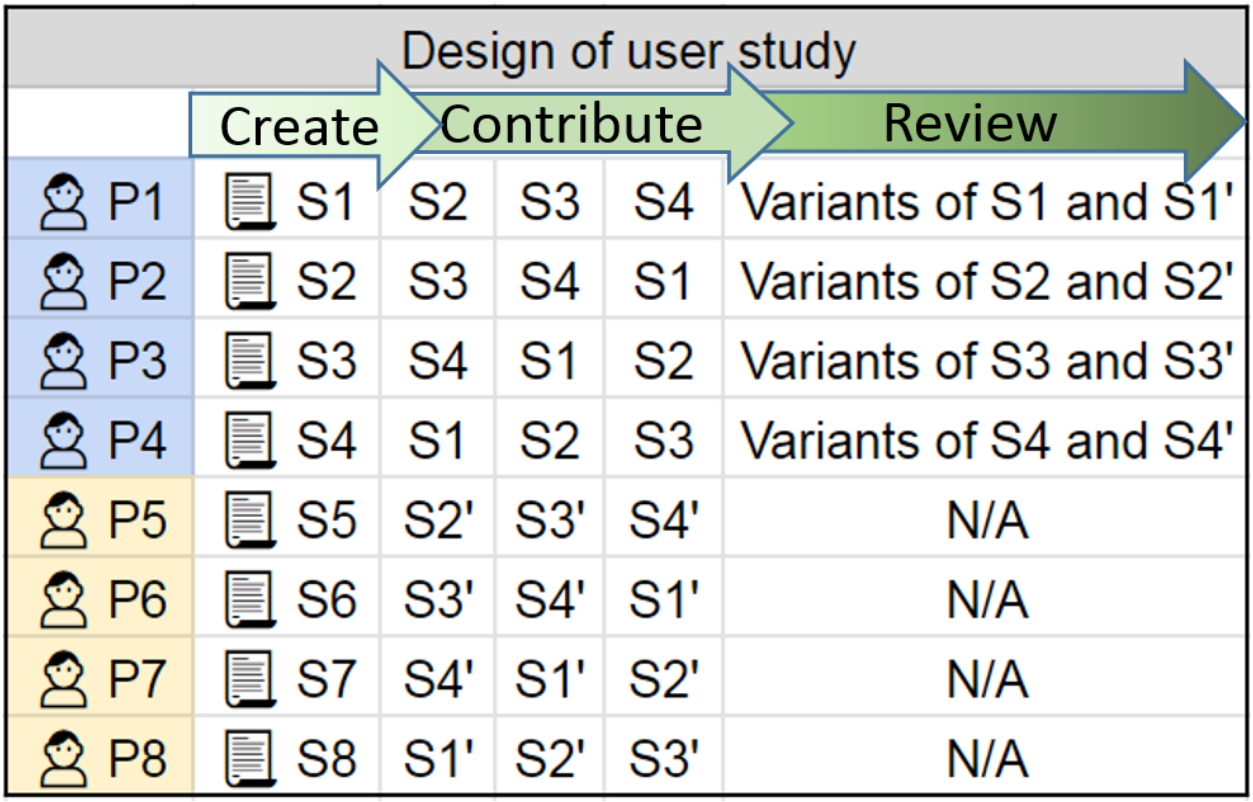

Figure 4.6. Evaluation process of user study. First, the participants in treatment (P1-P4) create storyboards (S1-S4), and participants in control (P5-P8) create S5-S8, respectively. Then they contribute to others' storyboards in Latin-square order. Si' means a panel-only copy of Si. To maintain the same context, storyboards S5-S8 are not used, while S1'-S4' are used instead. Finally, participants P1-P4 review all variants of their corresponding storyboards.

for 15 minutes each, as shown in Fig. 4.6. Finally, participants in T (P1-P4) will review all the alternatives of their storyboards.

This study process allows each participant to act as two roles (i.e., primary designers and peripheral contributors). In such case, participants in T can have better idea about the whole collaboration process so that they become more careful of writing their prompts. In the experiment, two participants revised their prompts after they started contributing and read others' prompts. Also, participants in C can have more experience with the system before they make contributions.

4.4.4 Apparatus

Both groups used the designer and contributor interface of CoStory to create panels (Fig. 4.3) and ideate on for requested clusters (Fig. 4.5). The difference is that the designer interface for group C was provided with curbed features, such as removing cluster-related features. Participants were allowed to use pen-and-paper to design their panels and upload their design to the system.

4.4.5 Data Collection

Participants in all groups completed a post-survey, in which they rated the usefulness and ease of the system on a 7-point Likert scale. In addition, contributors (both T and C) were asked to report their understanding level of the context and confidence level of contribution once they contributed to each storyboard. Primary designers (P1-P4) evaluated the diversity and usefulness of alternatives provided by contributors. Finally, designers (P1-P4) selected the "most preferable storyboard" among all panel-only and CoStory alternatives.

We also logged the click events involving key features of CoStory, such as checking alternatives, updating drawing, updating caption, and expanding a cluster.

4.4.6 Results and Discussion

Design topics and alternatives

Eight participants had eight different topics ranging from software to physical object, such as "An app to track the usage of cosmetic products", "Discover correlation in feature space with multi-dimensional data", and "Touch virtual objects in virtual reality".

Most participants preferred drawing with pen and paper, rather than drawing with mouse. Eight participants created eight initial storyboards, and then generated 24 alternatives in total, in which 12 in T (by P1-P4) and 12 alternatives in C by (P5-P8). Four participants in T created 1, 2, 2, and 3 clusters, respectively. But they chose to request help for 1 cluster, which implies that they used clusters to organize the design.

The length and complexity of the initial storyboards created by primary designers varied greatly: The shortest storyboard had 5 panels, whereas the longest had 11. The number of panels includes the panels within each cluster. After collaboration, the shortest storyboard had 5 panels, whereas the longest had 17 panels.

Below we discuss whether the hypotheses are supported by the results. To statistically evaluate the effect of CoStory on two groups, we used the Wilcoxon rank sum test to compare the differences. The choice of using wilcoxon rank sum test is because the dataset violates the normal distribution assumption of t-test.

H1: CoStory helps contributors understand the context better

Contributors self-reported whether they understood the context while they were creating an alternative. The wilcoxon rank sum test indicated that the scores reported by group T (Mdn=6.08, SD=0.90) was marginally significant than the scores by group C (Mdn=4.92, SD=1.56), U=40, p=0.053.

When the collaboration began, participants were able to see alternatives from previous contributors. All participants in T (i.e., P1-P4) reported that they checked all the existing alternatives before or during their authoring. This is confirmed by the log of user clicks on the tabs of each cluster. On average, each tab received 2.1 clicks, which means contributors in T usually check each alternative slightly more than twice.

In contrast, only one contributor in C (i.e., P6) reported that he went through all of the alternatives. The reasons why others did not check alternatives include: 1) each alternative seemed similar (we did not highlight the difference between two alternatives and thus looked similar at a glance), 2) they felt no need to check alternatives (the design topic may be too simple for them), and 3) they simply forgot to check.

This suggests that it's more efficient to manage alternatives on a cluster-level over storyboard-level, when the number of alternatives is large and the design topic is hard.

However, having easier access to alternatives may cause some side effects, such as design fixation [162]. We found that alternatives by T showed a clear pattern of iteration, while those by C showed more independent changes. The reason might be that contributors in

T tended to adopt ideas/elements from the existing alternatives due to easy access, while contributors in C tended to create their own as they did not read all of the alternatives. We may adopt some methods to reduce design fixation, such as incubation [163], or we may force them to avoid duplication.

H2: CoStory enhances contributors' confidence on their alternative

Contributors self-reported their confidence of the usefulness after each new alternative had been created. There is no significant difference between contributors in T (M=5.17, SD=1.34) and those in C (M=5.83, SD=0.72). This suggests that peripheral contributors feel confident about their contributions when given design context in two different ways.

H3: CoStory helps contributors meet the intent of designers better

Since contributors created alternatives following the prompts given by the primary designers. Primary designers were asked to evaluate whether each alternative 1) meets their intent, and 2) improves their design.

There is no significant difference between alternatives by T (M=5.58, SD=1.31) and by C (M=4.83, SD=2.17) on meeting the intent of primary designers. This suggests that using storyboard as the medium can communicate the design context and problem clearly, regardless of their drawing styles or fidelity.

However, there is a significant difference between the alternatives by T (M=5.67, SD=1.37) and by C (M=4.25, SD=1.81) on improving the design of primary designers (U=37, p¡0.05). This suggests that cluster-based storyboard is more efficient to receive design alternative. when the primary designers has a real design challenge and ask for help via cluster.

4.5 Limitations and Future Work

Although professional designers could be part of primary designers, we mainly evaluated the system with graduate students who have their own design problems. It is possible that professional designers may reflect different results during collaboration.

We asked all participants to contribute alternatives in the lab study, which may be different from the real life because participants are motivated differently. We measured the self-confidence of their contribution to imply their willingness of contributing more. In the future, we can ask primary designers to request assistance from their fellows and study the motivation aspect. Crowdsourcing may also be used as the peripheral contributors.

It would be interesting to study the influence of the level of asymmetry. For example, the collaboration efficiency may vary with different asymmetric degree/ratio of time spent on the design tasks.

4.6 Conclusion

This paper presents a system, CoStory, that implements *Synthesize-Request-Ideate* process for asymmetric collaboration. The system could augment storyboards with clusters, in which clusters form the basis of collaboration—request alternative design ideas and/or synthesize alternatives. By testing with 8 participants, we examined how the system could resolve some challenges of asymmetric collaboration. Specifically, contributors using all features of CoStory had higher confidence about their contribution, easier access to existing alternatives, and improved original design than contributors using traditional panel-only storyboards.

5. ADAPTUTAR: FLEXIBLE MACHINE TASKS FOR HUMAN WORKERS

This chapter presents the example of human-to-human task delegation, which is delegating customizable routine tasks to human workers through pre-recorded but adaptive tutorials. This chapter has adapted, updated, and rewritten content from a paper at CHI 2021 [10]. All uses of "we", "our", and "us" in this chapter refer to coauthors.

Modern manufacturing processes are in a state of flux, as they adapt to increasing demand for flexible and self-configuring production. This poses challenges for training workers to rapidly master new machine operations and processes, i.e. *machine task*s. Conventional in-person training is effective but requires time and effort of experts for each worker trained and not scalable. Recorded tutorials, such as video-based or augmented reality (AR), permit more efficient scaling. However, unlike in-person tutoring, existing recorded tutorials lack the ability to adapt to workers' diverse experiences and learning behaviors. We present *AdapTutAR*, an adaptive task tutoring system that enables experts to record *machine task* tutorials via embodied demonstration and train learners with different AR tutoring contents adapting to each user's characteristics. The adaptation is achieved by continually monitoring learners' tutorial-following status and adjusting the tutoring content on-the-fly and in-situ. The results of our user study evaluation have demonstrated that our adaptive system is more effective and preferable than the non-adaptive one.

5.1 Motivation and Contributions

Human workers are the most flexible part of the production process [164]. In the ongoing trend known as *Industry 4.0* [6], workers are expected to operate diverse machinery and other equipment in constantly changing working environments [5]. To meet these challenges, workers must rapidly master the machine operating procedures, referred as ***machine task***s. Numerous tutoring systems have been developed to facilitate the training [79], [165]–[168]. These novel tutoring systems show potential to eventually eliminate real-human one-on-one

tutoring, which will greatly lower the training cost and increase the scalability of workforce training.

Recorded tutorials permit more efficient scaling than live tutoring which requires in-person training. Prior studies [169]–[171] have compared tutoring effects between one-on-one live training and recorded tutorial-based training. Their results indicate that tutorial-based training is effective in efficient remote distribution and scalability, however, traditional one-on-one training has significant better training outcomes. This is because unlike a recorded tutorial which is mostly fixed and static once created, a live tutor can adapt to learners uncertainly during the training and adjust the tutoring content to achieve better results.

This concept of adaptation is particularly important in the *machine task* tutoring scenario, since workers are expected to be more versatile with various machine operations and processes, and the *machine task* environments are highly dynamic and spatial. Furthermore, each worker has their own different innate capability and strengths/weaknesses. In order to achieve better *machine task* skill transfer, it is crucial to design tutoring systems with capability of adapting to the ever changing working environment, as well as each individual worker.

In terms of tutoring presence, prior works have demonstrated the strength of humanoid avatar as a virtual representation of the user [79], for enhancing his/her bodily-expressive human-human communication. Besides, augmented reality naturally supports spatially and contextually aware instructions for interacting with the physical environment. Researchers have shown promises to use AR avatar as a virtual media for *machine task* tutoring applications [79], [98]. On the other hand, annotations [80], [88], [89] and animated components [90]–[92] have been widely used in prior AR research to provide tutoring content and guide users.

To this end, we present AdapTutAR, a *machine task* tutorial system with four kinds of AR elements that focuses on *adaptation*. Our system achieves adaptation by actively monitoring both the machine state as well as the user state during the tutoring process. We leverage the benefits of AR in spatial and contextual content visualization, and deep learning in object recognition as well as user activity recognition.

5.1.1 Contributions

The key contributions of this chapter are as follows:

- The design of the adaptation model that focuses on spatial and bodily visual presence for *machine task* tutoring.
- The design of corresponding features that enable adaptive tutoring in the recorded-tutorial environment based on *machine task* state and user activity recognition.
- The system implementation that achieves AR avatar tutorial recording, adaptive visualization and state recognition, and evaluation results from our user study

5.2 AR Tutoring Elements

In our work, we mainly focus on the tutoring of *machine task*s [79], in which a production process involves a compound sequence of *local*, *spatial*, and *body-coordinated* human-machine interactions [172], [173].

AdapTutAR aims to transfer the general process of *machine task*s to workers, such as what component (e.g., knob, lever) to operate, in what order, the exact state to change, and the expected outcome on each operation. Based on the prior work of AR visualization and the nature of *machine task*s, AdapTutAR chooses four types of tutoring elements to convey such sequential and logical knowledge to a learner (Figure 5.1).

1. **Avatar**. Since *machine task*s often involve spatial and body-coordinated human-machine interactions, the presence of AR avatar has shown benefits in *machine task* tutoring by improving learners' spatial attention and understanding of potential movement [79]. Specifically, the humanoid avatar can demonstrate the location of the interaction, the navigation path, and the body pose/gestures to accomplish a step.

2. **Animated component and arrow**. Given that each step of a *machine task* involves manipulating one or more machine components–such as knobs, buttons [174]–AdapTutAR animates the virtual representation of these interactable component(s). Nonetheless, when the animation is repeated in a loop, users may feel confused about

the actual direction of the animation (e.g., clockwise or counter-clockwise). Hence an arrow is added to clarify the direction. The animation and arrow help indicate how the component will look like when it is manipulated by a user.

3. ***Expectation of step***. When it comes to steps that require a user to set the machine to a specific state or parameter, it is often inadequate to convey the expected value by purely using animated components or arrows. To complement that, AdapTutAR shows expectation (e.g., the yellow text in Figure 5.1) right next to the animated component to indicate the expected value, such as "Set the printer head temperature to 500 F". The *expectation of step* has more formats than text. For some steps, AdapTutAR shows a virtual model as the expected value, such as a virtual car to be 3D printed or a tool to be used.

4. ***Subtask* description**. A *machine task* consists of multiple steps. Some consecutive steps may represent a cohesive subgoal, which is called a *subtask*. For example, a *subtask* "Replace the 3D printer head" involves loosing the safety lock, removing the existing printer head, picking up the expected one, installing it, and tightening the safety lock. A *subtask* description is shown at the top-left corner of a user's view to help the user build a higher-level understanding of the *machine task*.

5.3 Formative Study on AR Visualization

To inform the design of the adaptation model, we aimed to understand the performance of a user while learning a *machine task* using AR tutorials. A key objective was to elicit the users' preferences of exploiting the four tutoring elements and the requirements to the tutorial throughout the learning process.

5.3.1 Participants and Procedure

We recruited six participants (5 male, 1 female) aged 23 to 30. Four participants had experience with AR/VR systems while two did not. No participants had used AR/VR based tutoring systems before attending this study. (Participant: P)

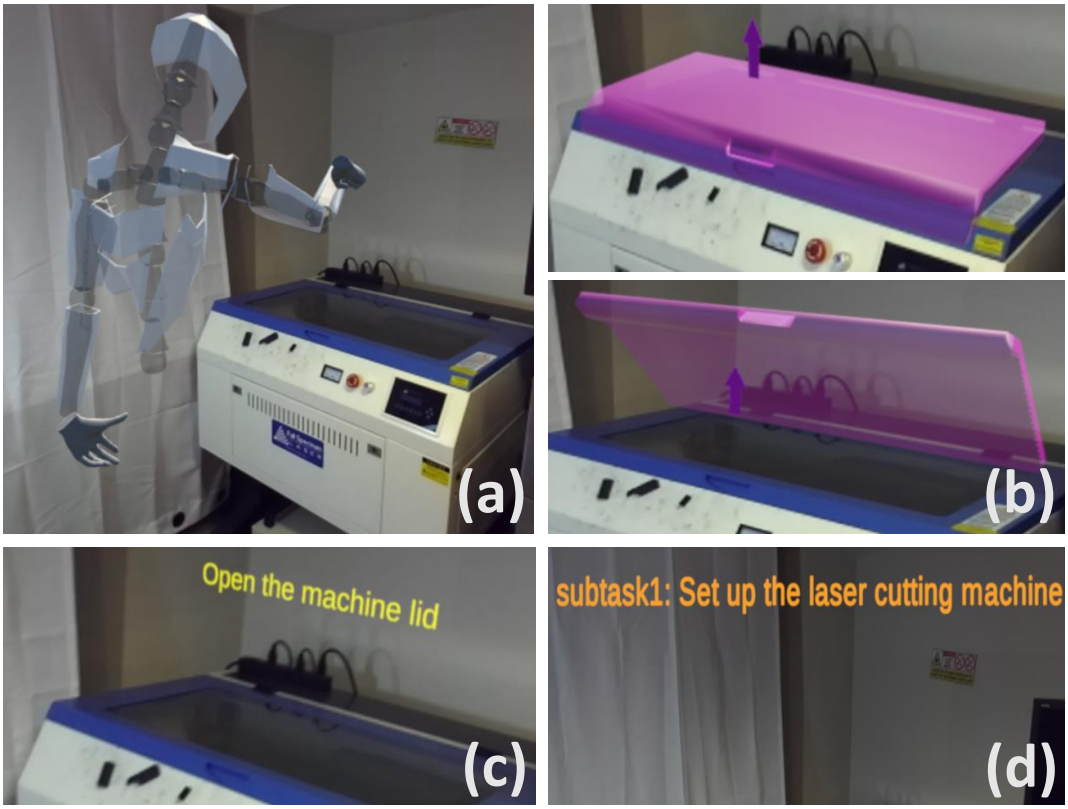

Figure 5.1. Tutoring elements: (a) Avatar, (b) Animated component and arrow, (c) *Expectation of step*, and (d) *Subtask* description.

We designed a laser cutting *machine task* consisting of interactions with physical interfaces and spatial navigation within AR environment. The participants were asked to learn the task using a pre-authored AR tutorial where all four tutoring elements were available in each step and the participants could manually toggle on/off any of them and browse along the steps using Oculus hand controllers [175]. Meanwhile, the participants were informed to learn the task in any way they felt efficient and comfortable by utilizing the tutoring elements, and the final goal was to remember and conduct the whole task without external assistance. The learning might go over for several times and end when a participant told the researcher he/she had mastered the whole task. The first-person view of the participants was screen-recorded and after the learning period, we interviewed the participants regarding the learning experience and our observations on their performance.

5.3.2 Findings

We analyzed the participant records in terms of the overall performance and the detailed actions in order to reveal the users' preference to the *machine task* learning. We analyzed 1) the pattern of step changing and tutorial following, 2) the timings when the participants toggled on/off the tutoring elements, 3) the combination of the tutoring elements at each step, and 4) the choices above at different learning stages. In addition, we analyzed the participants' bodily performance including their standing location, attention allocation, and so on. Finally, we distilled the higher-level design goals from our observations and the participant feedback.

Overall Learning Flow (F1)

Although the participants were able to navigate back and forth along the whole tutorial using the hand controllers, all six participants learned the task by following the tutorial step by step and repeated the whole task for multiple times. *"I think the order of these steps is critical to understanding the whole task. So, instead of mechanically remembering every single step, I learned them as an integrated story." (P6)* Moreover, all participants would only progress to the next step after ensuring the current step was completed correctly. *"I'd feel more confident if the system could tell me whether I did it correctly." (P3)* The performance and responses highlight that the adaptive tutoring model should be able to recognize the correctness of an operation in real-time and actively lead him/her to move forward in the task to ensure a fluent learning flow.

Combination of the Tutoring Elements (F2)

Overall, all participants agreed that the provided four tutoring elements were useful and sufficient for the learning. Yet, at different stages of the learning process, the participants chose different combinations of the tutoring elements. All six participants kept all four elements in the first trial, and went through every element in each step. *"The avatar was important when I first learned the task because it told me where should I focus on. And*

I also read the subtask description to briefly understand the purpose of the task." (P5) All participants agreed that as they were more practiced, the required tutoring elements shifted from specific demonstration to high-level description. *"After I knew those operations, the avatar was not that useful, but distracted me. So I turned it off." (P2)* These findings revealed that at different learning stages, the importance of each tutoring element varies. So the system should dynamically change the displaying tutoring elements as the learning progressed.

When to Show/Hide Tutoring Elements (F3)

We asked the participants to only keep the necessary tutoring elements while learning. First, we noticed that all toggle-off actions happened at the beginning of a step when a participant was clear he/she could master it. However, we observed that the toggle-on actions happened in more complicated scenarios. Compared to the toggle-off cases, before toggling on an element, the participants performed additional actions such as walking around the machine, attempting to operate an interface, correcting an operation rapidly and so on. *"Actually I was first trying to look around to find the next target, then if I couldn't, I turned on more." (P1)* This disclosed a need for the adaptation model first to understand the current state of the learner, then either provide more tutoring elements or reduce them. Additionally, we observed that timings when they turned on the tutoring elements varies at different learning progresses. *"When I almost learned everything, if I was stuck, I'd first recall the step, then turn on the elements. But initially, I didn't know much, so directly turned them on." (P4)* It unfolded another requirement for the model to adjust the timing to change tutoring elements accordingly.

Step-dependent Behaviors (F4)

For different steps, the participants selected different tutoring elements in the same trial. Meanwhile, when repeating the same step in different trials, and doing similar steps in the same trial, we observed that all participants gradually reduced the tutoring elements. *"The steps were different in some cases, so I'd like to use different elements. But there were some*

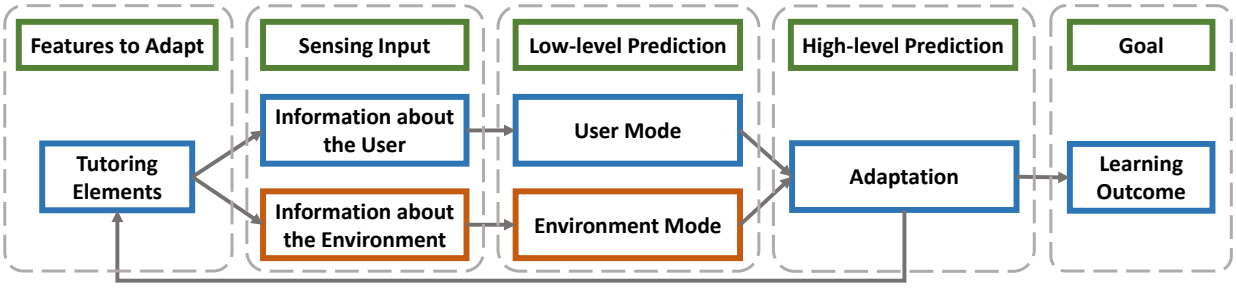

Figure 5.2. The Adaptation Model. Green boxes indicate the phases of adaptation.

similar ones, maybe the system could show me the previous choice." (P1) Meanwhile, when a step required spatial movements or complicated body-coordinated actions, the participants would spend more time for learning. *"Some steps were harder to learn, so I needed more time before I could turn off some tutoring elements." (P3)* Inspired by these findings, the model should also consider the nature of each step and the transition between any two steps.

5.4 Adaptation Model

To develop a tutoring system that can dynamically adapt the tutoring elements to match what a user actually needs, we organized the tutoring elements into five levels of details (LoDs), and further developed an adaptation model to adjust the LoDs in real-time (Figure 5.2). The key model includes four phases. Firstly, the system presents the tutorial at a given LoD. Secondly, the system collects the inputs from a user and environment in real time. Thirdly, the system performs low-level state recognition that recognizes the machine state, the user's basic mode, and region of interest (ROI). Finally, the system uses the low-level state to estimate the user's higher-level state, such as being stuck or not. Such estimation is transferred back to adapt the LoDs in the first phase.

5.4.1 Features to Adapt: Level Of Detail

The formative study confirms that each tutoring element serves a different role in conveying information, and further indicates that their necessity varies at different stages of the learning process *(F2)*. Therefore, we organize the tutoring elements into five levels of details (LoDs) as below.

- **LoD 5**: show all four tutoring elements.
- **LoD 4**: exclude avatar from LoD 5.
- **LoD 3**: exclude animated components and arrows from LoD 4. This essentially means it only shows expectation and *subtask* description.
- **LoD 2**: exclude step expectation from LoD 3, namely, just showing *subtask* description.
- **LoD 1**: show nothing.

When the LoD decreases, the tutoring elements are hiding gradually. The difficulty increases since there are fewer hints. In particular, for LoD 1 and 2, learners do not get direct hints about what component to operate nor what state to set to, which forces them to recall the details instead of being informed directly. On the other hand, learners who have gone through the same operations for multiple times may not need the detailed information provided in high LoDs.

As the first phase of each step, the system loads the historical LoD to determine what tutoring elements to present. If there is no historical data, the system shows the step with a default LoD (5).

5.4.2 Sensing Input

In this phase, AdapTutAR keeps collecting two categories of information: user and environment. User information includes the position and orientation of the AR headset as well as the first-person view of the user. Environment information includes the positions, orientations, and dimensions of the animated components and avatar.

5.4.3 Low-level State Recognition

In this phase, the inputs are used to recognize machine component state and user's basic state, which is further used to perform the higher-level state recognition in next phase.

Recognizing Machine Component State

The goal of machine state recognition is to detect what state the user has set the physical component to. This is required because the physical machine may not have sensor itself that could report the current state. In order to control the playback of the tutorial, its state change must be detected.

Prior works mostly focused on object detection and recognition (such as [176]), while a few focused on recognizing the specific states of an object [177]. However, these methods cannot be directly applied to our case where multiple identical objects may be visible on the same machine interface. For example, two knobs and two buttons are visible in Figure 5.5. The challenge is that after recognizing the states of these objects, the system needs to know which state belongs to which object. Inspired by LabelAR [178], AdapTutAR leverages the AR components that are aligned to the physical components. Besides the primary role of giving animations as instructions, AR components serve an additional role in providing the positions and dimensions in the world space, then AdapTutAR can obtain their 3D bounding boxes and further compute the 2D bounding boxes on the screen. Such bounding boxes help identify an object uniquely even when there are multiple identical objects within the scene. Finally, a Convolutional Neural Network (CNN) model is trained to recognize their object states within each bounding box. The detail is discussed in the Implementation section.

Recognizing User's Basic Mode

The goal of user state recognition is to identify what *basic mode* the user is in, including *static observation*, *navigation*, and *interaction*. To classify *interaction* mode, the key is to know whether a user touches the physical component or not. An approach similar to the aforementioned machine state recognition is used. For all visible machine components, the system crops out the camera images based on their bounding boxes, groups them into a batch, and predicts hand touching in parallel. If any component is touched by the user, the mode is classified as interaction. The user's state of *static observation* or *navigation* is predicted using the AR headset's position and orientation using a pretrained Support Vector Machine (SVM) model.

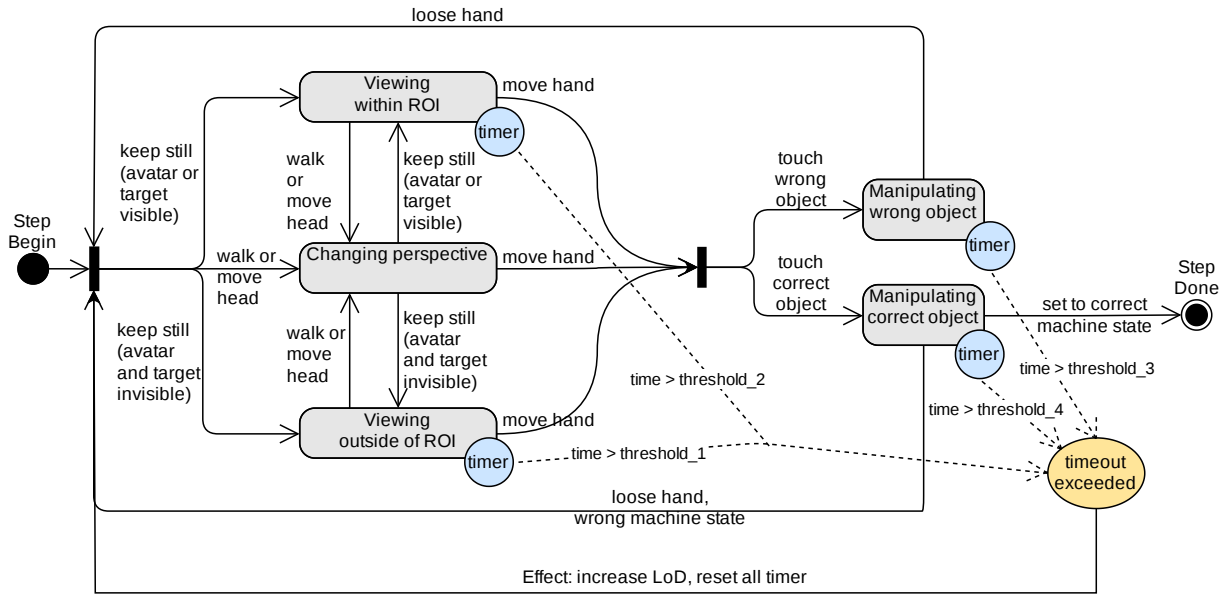

Figure 5.3. Inferred state of a user in a single step via Finite State Machine.

Classifying Region of Interest (ROI)

To classify whether a user is looking within ROI or outside of ROI, we first get the location of the target object(s) and avatar for a particular step, and compute whether they are visible by the user. This is essentially checking whether any of these objects is within the field of view (FOV) of the AR headset. If none of them is within FOV, the system classifies the user as looking outside of ROI; otherwise, within ROI.

5.4.4 Higher-level State Recognition and Adaptation

In this phase, the system estimates the user's state by forming a finite state machine and using the low-level recognition as inputs to drive the state transition.

Scenarios and States

We leverage the findings from the formative study *(F3)* and identify four scenarios in which users decided to turn on more tutoring elements, including (S1) unaware of the target, (S2) unaware of the operation, (S3) interact with wrong interface, and (S4) interact with the correct object for too long without setting to the expected state. The core of adaptation

is to estimate whether a user is currently under one of four scenarios and thus needs more tutoring content. To that end, we develop a finite state machine that takes the lower-level state recognition as input to help infer the states of a user (Figure 5.3). The four scenarios correspond to the exceptions of four higher-level states, including "Viewing outside of ROI", "Viewing within ROI", "Manipulating wrong object", and "Manipulating correct object". Due to their high correspondence, we also denote the four states as S1-S4, respectively. By monitoring how long a user stays in each state, the system determines whether the user enters one of the four scenarios.

State Transition

At the beginning of a step, the user immediately transits into one of three states: "Viewing outside of ROI", "Viewing within ROI", and "Changing perspective". The first state requires that the user is in static mode and looking outside of ROI, while the second state requires that the user is in static mode and looking within ROI. When the user is in navigation mode (e.g., walk or move head), their state will be transited to "Changing perspective". Once the user pauses walking or moving head, the state will be transit into the first or second state accordingly.

When the user touches a machine component, the user state transits into one of two states: "Manipulating wrong object" and "Manipulating correct object". This transition depends on whether or not the touched component is the expected one in the current step. While the user is manipulating the correct object, our system keeps recognizing the machine state and comparing it to the expected one. If matched, the current step is done. If not and the user stops manipulating, the user state transits back to one of three states related to viewing and changing perspective.

Each state has an independent timer which resets at the beginning of each step. When a user transits from one state to another, the timer of the original state pauses while the timer of the new state starts ticking. If a user remains in one state for too long (i.e., accumulated time > threshold), the system estimates that a user may be stuck in one of four scenarios. For example, if a user stays in "Viewing outside of ROI" state for time longer than $threshold_1$,

the system estimates that the user is "unaware of the target" *(S1)*. Likewise, if a user stays in "Viewing within ROI" state for time longer than $threshold_2$, the user is inferred to be "unaware of the operation" *(S2)*. Consequently, an event is triggered to increase the level of detail (LoD) and reset all timers. The calculation of thresholds can be found in section 5.4.4.

Finally, if a user completes a step, the LoD is decreased by 1 *(F3)* and saved into user's profile for future reference. Note that the LoD data is tied to a component, not to a step. This is because a component may be involved in multiple steps of the same tutorial or be shared in different tutorials that involve the same machine. Therefore, binding LoD data to a component rather than a step supports better reuse of the user's learning record. As mentioned in the first phase earlier, the system loads the historical LoD at the beginning of each step. Here, it only loads the most recent LoD of the component, and ignores the older LoD record(s) if any.

In summary, the adaptation model leverages both the information from the historical record and the real-time inputs, which is a combination of macro and micro-adaptive approaches [58], [59].

Timer Threshold

The key is to find a proper threshold for each state/scenario. The first empirical observation is that the threshold should refer to the actual time spent by the expert in each step. A more complicated step takes longer time than an easy step so that the former should have a larger threshold than the latter. Based on section 5.4.3, the total time spent in step i can be further decomposed into the time spent in observation ($observeTime_i$), navigation ($navigateTime_i$), and interaction ($interactTime_i$). Moreover, the threshold of each state should refer to the most relevant type of time. For example, the states related to manipulating objects (S3 and S4) should refer to the interaction time, while the states re-

lated to viewing (S1 and S2) should refer to observation and navigation time. Therefore, let $ReferenceTime_{\mathrm{i}}$ denote the reference time of states (S1-S4) in step i:

$$\mathbf{ReferenceTime}_{\mathrm{i}} = \begin{bmatrix} t_{\mathrm{i}}^{s1} \\ t_{\mathrm{i}}^{s2} \\ t_{\mathrm{i}}^{s3} \\ t_{\mathrm{i}}^{s4} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} observeTime_{\mathrm{i}} \\ navigateTime_{\mathrm{i}} \\ interactTime_{\mathrm{i}} \end{bmatrix} \tag{5.1}$$

For example, the reference time of manipulating correct object (S4) in step i is $t_{\mathrm{i}}^{s4} = interactTime_{\mathrm{i}}$. Also, when a learner is manipulating a wrong object (S3), it is supposed to have a smaller threshold so that the hints can be shown faster. Therefore, an empirical value (0.5) is chosen so that $t_{\mathrm{i}}^{s3} = 0.5 \times interactTime_{\mathrm{i}}$.

Lastly, the current LoD matters. A larger LoD implies that a user is less proficient in this step so that the system should tolerate a larger threshold. Let $Threshold_{\mathrm{i}}$ denote the thresholds for states S1-S4 in step i:

$$\mathbf{Threshold}_{\mathrm{i}} = \begin{bmatrix} threshold_{\mathrm{i}}^{s1} \\ threshold_{\mathrm{i}}^{s2} \\ threshold_{\mathrm{i}}^{s3} \\ threshold_{\mathrm{i}}^{s4} \end{bmatrix} = f(LoD_{\mathrm{i}}) \times \mathbf{ReferenceTime}_{\mathrm{i}} \tag{5.2}$$

where $f(x)$ is a factor that scales the reference time based on the current LoD. In this project, we take:

$$f(LoD) = 1 + log_5(LoD) \tag{5.3}$$

When LoD=5, $f(x)$ is 2, which means the threshold is twice of the reference time. This is because at LoD 5, a learner is inferred as a novice so that they may spend $1 \times referenceTime$ in purely watching the tutorial and $1 \times referenceTime$ in following the tutorial. On the other hand, when LoD=1, $f(x)$ is 1 because the system infers the user as proficient to this step and tolerates the same time as the reference time. Using a *log* function rather than a linear

function is to make the threshold decrease slower in large LoD (4 and 5) and faster in small LoD (1 and 2).

5.5 Adaptive tutoring system

To support effective apprenticeship for *machine task*s in workshops or factories, we designed and implemented AdapTutAR. AdapTutAR is an AR-based authoring and tutoring system that enables an expert to record a tutorial that can be learned by different workers in an adaptive way.

5.5.1 Workflow Illustration (Overview)

AdapTutAR consists of three modes: 1) Authoring Mode in which an expert can record a tutorial; 2) Edit Mode in which the expert can edit the tutorial; and 3) Learning Mode in which workers can learn the tutorial.

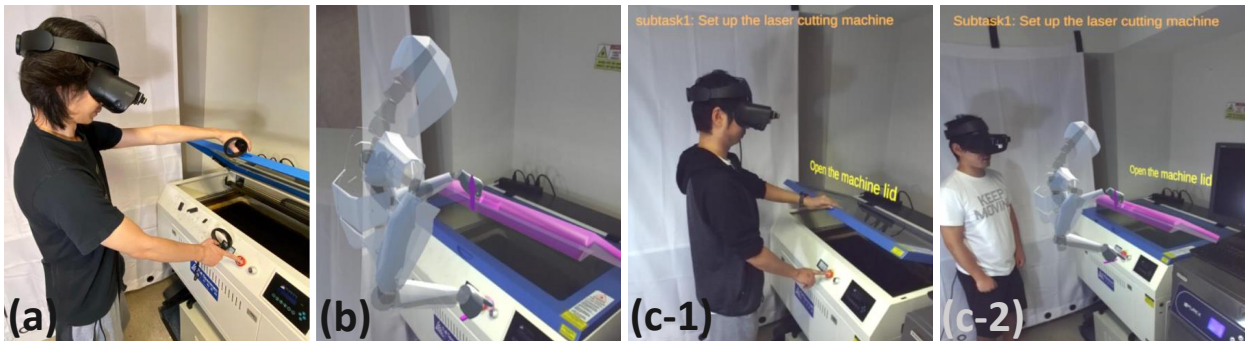

Figure 5.4. An overview of AdapTutAR workflow. (a) An expert records a tutorial. (b) The tutorial is represented as an avatar and animated components with arrows. The expert can edit the tutorial by adding *subtask* description and *expectation of step*. c) The same tutorial is adaptively shown to two learners. The learner in (c-1) is given less tutoring contents than the learner in (c-2) due to the difference of their experience and learning progress.

Prerequisite: Setup Training Environment

Before an expert can record a tutorial for a machine, they need to set up the training environment first. This requires that a machine has a digital copy aligned with the physical

counterparts, and also the machine state recognition model has been trained. However, since the focus of this chapter is not about setting up the environment but the adaptive tutoring within the environment, for simplicity, we assume the experts are given a machine that has already been set up. Without losing generality, an expert can also use the following process to set up a new training environment. First, the expert uses the hand-held controllers to align the virtual components with their physical counterparts in the AR world. Secondly, they can follow the pipeline mentioned in the later subsection 5.5.2 to collect a dataset for machine state recognition. Once enough data is collected, our system can fine-tune the CNN model with the dataset. The dataset only need to be collected once for one type of machine.

Authoring Mode

Tutorials are authored using an natural embodied movement, where the system records the expert's body motion by tracking the position and orientation of the AR headset and two hand-held controllers (Figure 5.4a). In addition, the expert can manipulate the virtual component(s) through different gestures of virtual hands powered by the controllers. During the operating process, the human motion and the 6 DoF poses of the virtual components are recorded. The recorded human motion will be represented as avatars while the recorded pose sequences of the virtual components will become AR animations. To partition the entire tutorial into steps, the expert needs to explicitly starts and stops recording each step by pressing the joystick on the controllers.

Editing Mode

Once all steps are recorded, the expert can enter the Edit Mode to label descriptions. The expert can pick a step to add expectation or select several consecutive steps to add a *subtask* description. To add a subtask description, the expert can enter a short sentence via virtual keyboard. To add an expectation, the expert first creates the text in a similar way and then uses controllers to anchor it to the proper position in the environment.

Learning Mode

A learner wears the AR headset and starts to follow the first step of the tutorial without hand-held controllers (Figure 5.4c). Four types of tutoring elements may be shown/hidden dependent on the current LoD. By default, a user starts with LoD 5 so that they are given all elements to guide how to operate. As a learner may need to repeat the tutorial for multiple trials before comprehending it, the system keeps adapting the tutoring content for each step based on their historical learning progress and the current behavior. This is achieved by the aforementioned Adaptation Model that is running in the background. It also monitors if the learner has set the component to the expected state. The tutoring elements remains until the learner operates correctly *(F1)*.

5.5.2 Implementation

System Hardware and Software Setup

The see-through AR platform is built by attaching a stereo camera (ZED Dual 4MP, 720p) in front of a VR headset (Oculus Rift [175]). Four external Oculus IR-LED sensors are used to track the human body motion with an effective area of 3 x 3m. Two Oculus Touch Controllers enable authoring by an expert. The main AR interfaces are developed with Unity3D [179], and the predictions are made with a backend server running aiohttp[1] web framework in Python. The backend server loads the models trained by Tensorflow (v2.1) and SVM. Both Unity3D and backend server run on the same PC (Intel Core i7-9700K 3.60GHz CPU, 32GB RAM, NVIDIA GeForce RTX 2070). The stereo camera provides built-in streaming functionality that can be accessed by the server. Unity3D sends data to the server via Socket.IO, including the objects to be tracked, their bounding boxes, and the positional data of the headset. In return, the server sends the predicted machine state and user state back to Unity3D via Socket.IO.

[1] ↑https://docs.aiohttp.org/

Recognizing Machine Component State

As mentioned earlier, we leverage the bounding boxes of virtual components to uniquely identify physical components. As there are different types of machine components, we developed an efficient pipeline to collect dataset based on video streaming and bounding boxes. Note that if some components are identical, users only need to collect dataset based on their type (e.g., knob, lever), rather than individual components. First, a user sets a physical component to a specific state or sets multiple components to specific states, such as "1" for knob in Figure 5.5. Then the user selects their virtual counterpart(s) in our system, sets the state(s) to match the physical one(s), and starts video streaming of ZED camera. The video stream is automatically cropped into RGB-D images based on the bounding boxes and also labelled with the current states and types. To make the dataset comprehensive, the user needs to look at the object(s) from various heights, places, and angles. Such process can be repeated to cover the remaining states of the component(s) as well as other interactable components of the machine.

The collected images are used to train a CNN model for state recognition, as shown in Figure 5.5. First, each image is resized to 100x100x3. Then data augmentation is done by adding random hue (max_delta=.2), saturation (0.1∼2.0), contrast (0.3∼1.0), and brightness (max_delta=.5). The feature extraction of the CNN model is based on MobileNetV2 [180]. After feature extraction, the output size of the base model is 4x4x1280, which follows by layers of MaxPooling2D, Flatten, two fully connected layers (units=1024 and 64) with Relu activation, Dropout(0.5), and a fully connected layer (units=number_of_component_types) with "softmax" as activation. The loss function is "tf.keras.losses.SparseCategoricalCrossentropy" and the optimizer is SGD.

Recognizing Hand Touching

This is similar to the machine state recognition above. The difference here is that instead of setting the state of a component, a user needs to act two states on each component type, including "hand not touching" and "hand touching". Overall, 130k images were collected

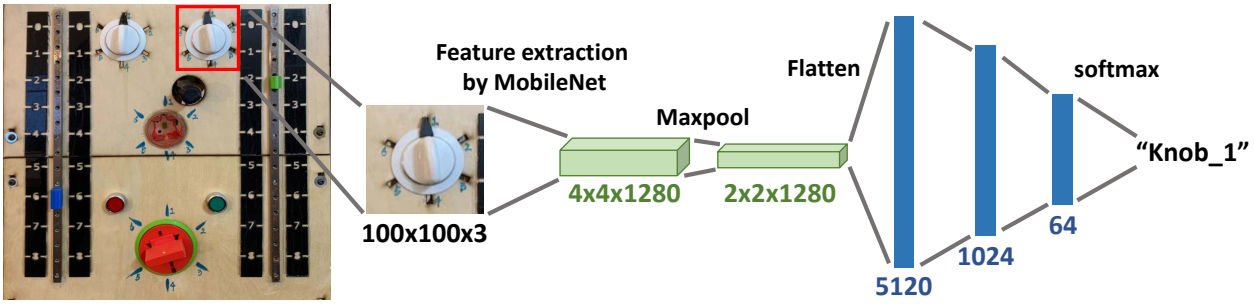

Figure 5.5. CNN model for machine state prediction based on bounding boxes.

for 9 component types by four volunteers. Finally, the images cropped out by the bounding boxes were used to train a different CNN model.

Recognizing Static Observation and Navigation

We used Support Vector Machines (SVMs) to recognize the static observation and navigation, which have demonstrated high performance when applied to human and animal activity recognition tasks [181], [182]. The feature vector is computed by taking the magnitude difference between kth and 0th frames in a window ($k = 0, ..., windowSize$) for each user head position $\mathbb{R}^3$ and orientation $\mathbb{R}^4$ vector. If the absolute magnitude difference is greater than the threshold, features describing changes in the head position and orientation are set to true. Optimal magnitude thresholds were determined by grid search. Three volunteers generated 90 samples for these two states in which each sample lasted about 10-20s. By performing a grid search, our features were extracted using a window size of 1.3s with an overlap of 0.56s (stride).

5.5.3 Preliminary System Evaluation

AdapTutAR relies on the capabilities of the low-level state recognition. To evaluate these capabilities, we conducted a preliminary system evaluation.

Accuracy of Machine State Recognition

We built a mockup machine with 9 types of machine components to train and test the model. Figure 5.5 shows one side of the mockup machine. By using the pipeline in section

5.5.2, three volunteers collected 171K images for 9 component types with 31 distinct states. For example, a knob has 6 states, a key hole has two states (inserted or not), and a switch has two states (on or off). Given that the video streaming is 60 FPS and the user keeps changing the view angles, we save one image every 8 frames to avoid identical images. The training took 10 hours on an NVIDIA GeForce RTX 2070. Before the test, each component of the mockup machine was set to a particular state that was entered into the system as ground truth. During the test, the tester wore the AR headset and looked at the component from different angles for approximately 3 seconds. The video stream was cropped out based on the bounding boxes and sent to the trained model directly. Each batch of images took about 0.11s in prediction. Then the predicted states were compared with the ground truth.

Three researchers participated in the test and produced about 2k results for all the 31 states of 9 component types. The overall classification accuracy was 89.1%. Specifically, the levers and knobs produced small errors (95.5% accuracy) while sliders and key holes produced larger errors (85.3% accuracy). In general, the system can satisfy the requirements of machine state recognition.

Accuracy of Hand Touching

A similar approach was used to test the accuracy of hand touching. The difference was that for each component type, the tester's left and right hands alternatively touched the component. Three researchers participated in the test and produced 913 results. The overall classification accuracy was 93.4%, which validated the feasibility of our system in accurately recognizing hand touching on machine interfaces. A typical error happened when the hand was close to, but had not touched the component. Such scenario was often mis-classified as hand touched. Fortunately, such error did not greatly affect the adaptation model because if a user moved the hand close to a component, it implied that the user intended to touch it.

Accuracy of Classifying Static Observation and Navigation

During the test, participants performed three actions: 1) standing still and moving head slowly; 2) standing still and moving head drastically; and 3) walking. The first action should

be classified as static observation and the last two actions should be navigation. During the test, three researchers performed each action for roughly 5 seconds, and repeated for 3 times. By splitting the sequences by the window size (1.3s) and stride (0.56s), there were 241 predictions. The overall accuracy was 92.1%. The errors were partly due to the transition from one state to another. This result indicates that the system can detect the user's basic mode accurately.

5.6 User Study

Complying with the requirements of social distancing for COVID-19, we conducted a 2-session remote user study in Virtual Reality (VR).

5.6.1 Study Setup

Since the remote users had no access to the real machines, we built a virtual multi-function machine that enables 3D printing and painting (Figure 5.6), which was inspired by prior works [99], [183] that validated key features of AR systems in VR. The virtual machine and the two tooling tables were located within a $3m \times 4m$ virtual space. The VR application was sent to the users and the user study was completed using their own VR devices.

During the user study, the users learned a 28-step plastic toy fabrication task using the virtual machine (Table 5.1). The users had to set the machine parameters using knobs, buttons, switchers or sliders (*local tasks* e.g. step 1 to 5), to deliver correct raw materials (*spatial tasks*, e.g. step 7, 8, 10, 11) and to assemble the tools properly (*body-coordinate tasks*, e.g. step 8, 19, 21). The adjacent steps that served a high-level purpose were grouped into one *subtask*, e.g. the purpose of step 6 to 9 was tooling installation. Some of the steps (e.g. step 8) could be accomplished only if some previous steps (e.g. step 6) were conducted correctly. The users were interacted with the machine using VR hand controllers. Note that in the VR simulation, we directly used the collision between the VR controllers and the machine components to detect the interactions rather than the image classification technique in AR environment. Consequently, the accuracy of machine state and hand touching recognition

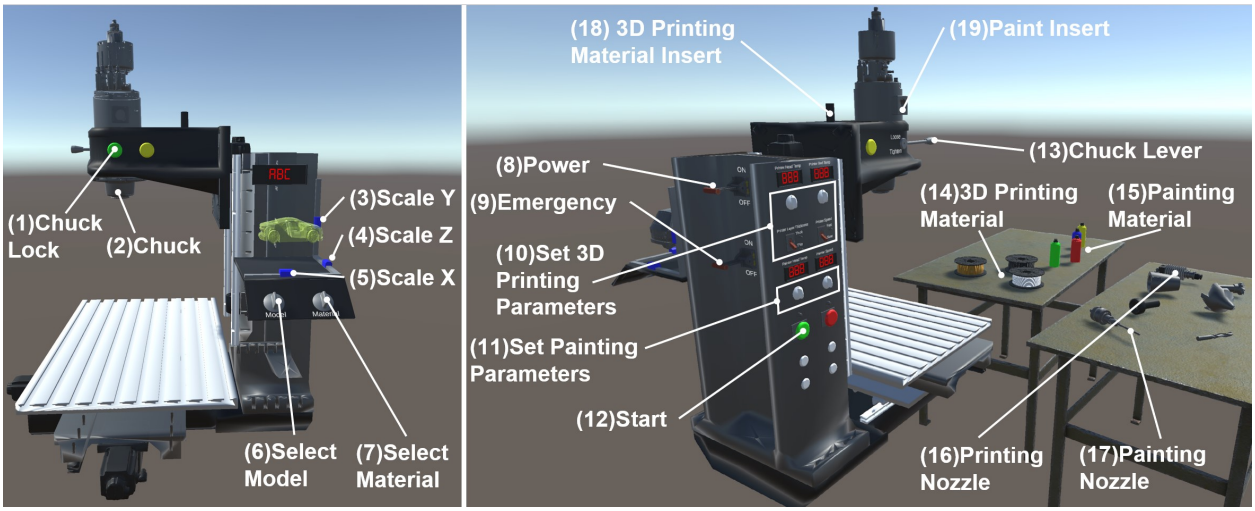

Figure 5.6. The VR environment of the user study, including a multi-function machine and several tools and materials for 3D printing and painting.

in VR reaches 100%, which is different from the aforementioned accuracy in AR. We discuss the limitations and mitigation in the next section.

We recruited 24 users (19 male, 5 female, aged 18 to 35) to our remote user study. 19 out of 24 users have engineering background while the other 5 have science background. 11 users have AR/VR experience and 15 users have hands-on machine operation experience. Nine users owned VR devices and shared with their friends or roommates for our user study. Specifically, 14 users used Oculus Rift [175] while 10 users used Oculus Rift S [175]. None of the users had experience with our system before. All the users were compensated with a $20 gift card for the 1.5-hour user study.

5.6.2 Study Design

We evaluated the benefits and limitations of our adaptation model by comparing an adaptive VR tutorial (noted as *adaptive*) with a similar VR tutorial that had no adaptive features (noted as *non-adaptive*) through a within-subject study. In *adaptive* tutorials, the VR tutorial elements were adjusted following the strategy proposed in section ADAPTATION MODEL, while in *non-adaptive* tutorials, the VR tutorial elements were displayed at a fixed LoD of 5 (none of the tutorials elements were hidden). The users were requested

to learn two *machine task*s (Table 5.1) in two sessions with the two types of AR tutorials respectively.

Both the two *machine task*s were plastic toy fabrication tasks but differed in fabricated models (step 1), materials (step 2, 10), sizes (step 3 to 5), colors (step 11) and machine parameters (step 8 to 10, 26, 27) to avoid the users from remembering the task in the last session. However, both tasks shared similar machine interfaces and step orders. To counterbalance the learning effects, we separated the users into two groups randomly. Specifically, 12 users followed the *adaptive* tutorial in session 1 and the *non-adaptive* tutorial in session 2. In contrast, the other 12 users followed the *non-adaptive* tutorial in session 1 and the *adaptive* tutorial in session 2. The users were not informed with the tutorial conditions.

Each session contains a tutoring section and a testing section. In tutoring section, the users had up to 30 minutes to learn the task by following the tutorial and performing machine interactions. Both of the tutorials were able to proceed to the next step automatically when the user conducted a step correctly. The tutorial repeated from the beginning when a user completed the last step. After the users claimed they had understood and remembered the task, or reached the time limitation, the researcher terminated the tutoring section. Then the users entered the testing section after a 3-minute rest. In the testing section, the users completed the task with all the AR tutoring elements hidden.

In each session, users repeated the tutorial multiple times before they entered the testing section. After session 1, users would change from *novice*, who had little experience in the machine and environment, to *proficient*, who could clearly describe the task purpose and complete the required operations without external hints. Note that those proficient users were not considered as proficient in all machine tasks but only in the second task, provided that the second task shared similar machine operations and step orders with the first task. Thus, we were able to evaluate the user performance under different conditions, e.g., *novice* with adaptive, *proficient* with adaptive, *novice* with non-adaptive, and so on.

Table 5.1. Tutorial used in the user study that involves 3D printing and painting. Widget # is referring to Figure 5.6.

Subtask	Step		Widget
Set output model to a Jeep vehicle with PLA material	1	Select model	6
	2	Change material	7
	3	Change scale X	5
	4	Change scale Y	3
	5	Change scale Z	4
Change tool head to printer head	6	Loose chuck lever	13
	7	Pick up printer nozzle	16
	8	Install printer nozzle	1,2,16
	9	Tighten chuck lever	13
Install PLA 3d printing material	10	Pick up printer filament	14
cont.	11	Install printer filament	18
Set parameters for PLA 3D printing material	12	Turn on machine	8
	13	Set head temperature	10
	14	Set bed temperature	10
	15	Set layer thickness	10
	16	Press start button	12
Change tool head to painter head	17	Turn off machine	8
	18	Loose chuck lever	13
	19	Remove printer nozzle	1,2,16
	20	Pick up painter nozzle	17
cont.	21	Install painter nozzle	1,2,17
	22	Tighten chuck lever	13
Install painting material	23	Pick up painting material	15
	24	Install painting material	19
Set parameters for painting	25	Turn on machine	8
	26	Set head temperature	11
	27	Set painting time	11
	28	Press start button	12

5.6.3 Data Collection

During each tutoring section, we recorded the total time that the user took and the times that the tutorial had repeated. To better understand the user's behavior, we also recorded the LoD of each step and the reason if the LoD changed (e.g. hesitate for too long, manipulate the wrong object). After the tutoring section, we let the users to evaluate their learning progress using 5-point Likert-type questions (Figure 5.7 left). For testing section, we used the time that the user consumed as well as the number of mistakes to quantify the learning outcome. After the two sessions, the users voted for their favourite AR tutorial. Further, users rated their subjective feelings about the adaptive features of AdapTutAR using another 5-point Likert-type questions. Finally, A conversational interview was conducted and recorded regarding the the reason why the users preferred an AR tutorial and the insights to improve the adaptive features of AdapTutAR. Additionally, the users' first personal view in VR was recorded for further analysis.

5.6.4 Results

In this subsection, we present objective performance and the subjective ratings of this study.

Self Rating on Learning Experience

After each tutorial section, the users rated their learning experience of performing the *machine task* using the 5-Point Likert Scale questionnaire (Figure 5.7). We separated the users based on whether they were novice or proficient and which AR tutorial that they just used. All users reported that they could operate the machine correctly ($M = 4.71, SD = 0.50$) and understand how the machine works ($M = 4.86, SD = 0.37$). Meanwhile, all users were generally confident to finish the task without hints ($M = 4.5, SD = 0.69$), and felt they remembered each step of the task ($M = 4.58, SD = 0.73$). While the proficient users who had used the *adaptive* tutorial rated themselves slightly higher then other users, an one-way ANOVA performed on each group of the ratings showed that there was no significant difference in the ratings regarding "Accuracy" ($p = 0.10$), "Understanding"($p = 0.12$), "Memorization"($p = 0.46$), and "Confidence"($p = 0.27$). In spite of the AR tutorials and the background, all users reported that they had mastered that *machine task*.

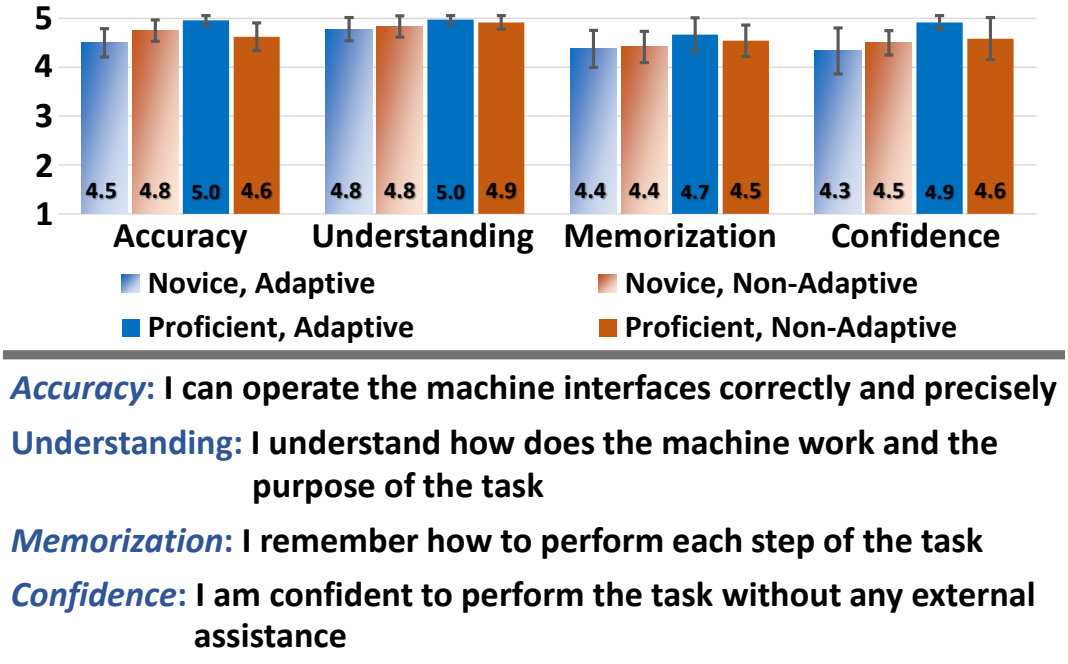

Figure 5.7. The users' self-evaluation of the learning progress.

Objective Performance

We compared the performance of the novice users and proficient users respectively due to the cognitive gap between the novice and the proficient regarding the *machine task*. A t-Test was performed on each pair of the data. Regarding the learning time, the novice users who used the *adaptive* tutorial took significantly more time ($M = 23.0, SD = 7.3$) in tutoring section than the ones who used the *non-adaptive* tutorial ($M = 16.5, SD = 5.7, p = 0.023$). A similar conclusion could also be drawn from the proficient users (*adaptive* $M = 10.1, SD = 2.9$, *non-adaptive* $M = 7.6, SD = 1.7$, $p = 0.029$). The novice users who were using the *adaptive* tutorial required more repeats of the tutorial ($M = 4.1, SD = 1.0$) than the ones with the *non-adaptive* tutorial ($M = 3.3, SD = 1.2, p = 0.017$). The proficient users required similar times of repetition (*adaptive* $M = 2.6, SD = 1.2$, *non-adaptive* $M = 2.3, SD = 0.5$, $p = 0.50 > 0.05$). Consequently, the users who used the *adaptive* tutorials needed more time (novice $M = 7.2, SD = 2.2$, proficient $M = 3.6, SD = 1.0$) than the user with the *non-adaptive* tutorial to go through the tutorial for one time (novice $M = 4.5, SD = 1.7$, proficient $M = 2.7, SD = 1.0$). In the testing section, novice users using the *adaptive* tutorial took slightly shorter time ($M = 3.2, SD = 0.6$) then the novice users with the *non-adaptive* tutorial ($M = 3.9, SD = 1.3, p = 0.13 > 0.05$), while the proficient users took approximately same time (*adaptive* $M = 2.8, SD = 0.4$, *non-adaptive* $M = 3.2, SD = 0.9$, $p = 0.27 > 0.05$). Notably, the novice users who used the *non-adaptive* tutorial made more mistakes ($M = 2.17, SD = 1.52$) than the ones with the *adaptive* tutorial ($M = 1.00, SD = 1.04, p = 0.039$). The difference between the proficient users was not obvious since they were making few mistakes (*adaptive* $M = 0.41, SD = 0.66$, *non-adaptive* $M = 0.75, SD = 0.75, p = 0.26 > 0.05$). The results are presented in Figure 5.8.

User Preference Vote

The users voted for their favorite AR tutorial based on their overall experience as well as considering the training efficiency, the learner's understanding of the task, and the comfort of the learning experience respectively (The Figure 5.9 Left). Overall, most of the users preferred the *adaptive* tutorial (21 out of 24). Meanwhile, the users also agreed that

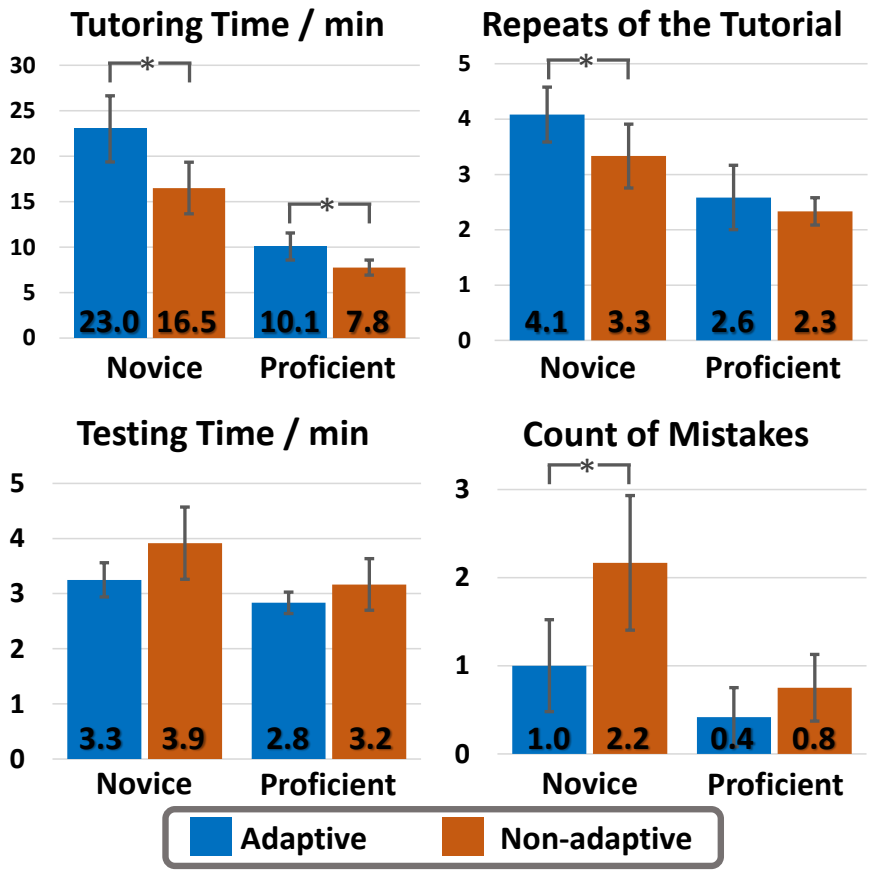

Figure 5.8. Objective performance during the tutoring and testing sections for novice and proficient participants. ($* = p < .05$).

the *adaptive* tutorial delivered a more comfortable learning experience (21 out of 24). In terms of the efficiency and understanding, a little more users (about one-third) chose the *non-adaptive* tutorial, while the majority of the users (about two-thirds) still preferred the *adaptive* tutorial.

Subjective Rating

The Likert-type ratings regarding the adaptive features collected from the user study are shown in Figure 5.9 right. In general, the users agreed that the *adaptive* tutorial provided appropriate information in time (Q7: $M = 3.8, SD = 0.9$, Q8: $M = 4.8, SD = 0.4$). "*The adaptive tutorial showed the tutorial elements that met my requirements. The AR avatar is the most helpful at first because it was straightforward and intuitive. Later I found the subtask*

description was more helpful because it reminded me what to do next." (P16) Meanwhile, the users acknowledged that the *adaptive* tutorial can properly hide the redundant information that was not needed and consequently the AR visualization is clear and non-distractive, especially compared with the *non-adaptive* tutorial. (Q3: $M = 4.2, SD = 1.0$, Q4: $M = 4.3, SD = 0.8$) "*Since I already went through the last session, (as a proficient user) I thought I didn't need that much tutoring element, and the adaptive system hided those not needed." (P7)*,"*With the adaptive system, my view was clearer because there were less AR elements distracting me." (P6)* The users also appreciated the adaptive feature which helped them to understand the task (Q5: $M = 4.8, SD = 0.4$, Q6: $M = 4.7, SD = 0.5$). "*After the avatar was hidden, I started to pay attention to the descriptions and got to understand the logic behind each machine operation." (P1)*,"*When the adaptive tutorial let me do it by myself, my brain was active and trying to understand the logic between the steps." (P15)* Moreover, it was receptive by the users that the *adaptive* tutorial made them better remember the tasks (Q1: $M = 4.5, SD = 1.0$, Q2: $M = 3.8, SD = 1.3$). "*The adaptive tutorial was gradually increasing the difficulty, which force me to remember the steps." (P14)*, "*Although I wasn't sure about the parameters of that step, I tried to perform it by myself and succeeded. This experience gave me a impression of that step." (P16)*

5.7 Discussion, Limitations and Future Work

In this section we discuss the primary results of the study and also provide design recommendations and insights for future adaptive tutoring systems.

Design of LoD. The design of LoD was first inspired by the formative study finding *(F2)*, and further proved to be receptive through the user study. The users agreed that the arrangement of the LoD fullfilled the needs at different learning stages. Yet, some users raised an interesting point. *"First, I pretty like the decrease of the tutoring elements as I learned more. But I wonder if the system could change the performance of the avatar or the animation according to my performance." (P14)* It reminds us that our adaptation model could also drive the modification of the tutoring elements in each LoD, not just hide or show.

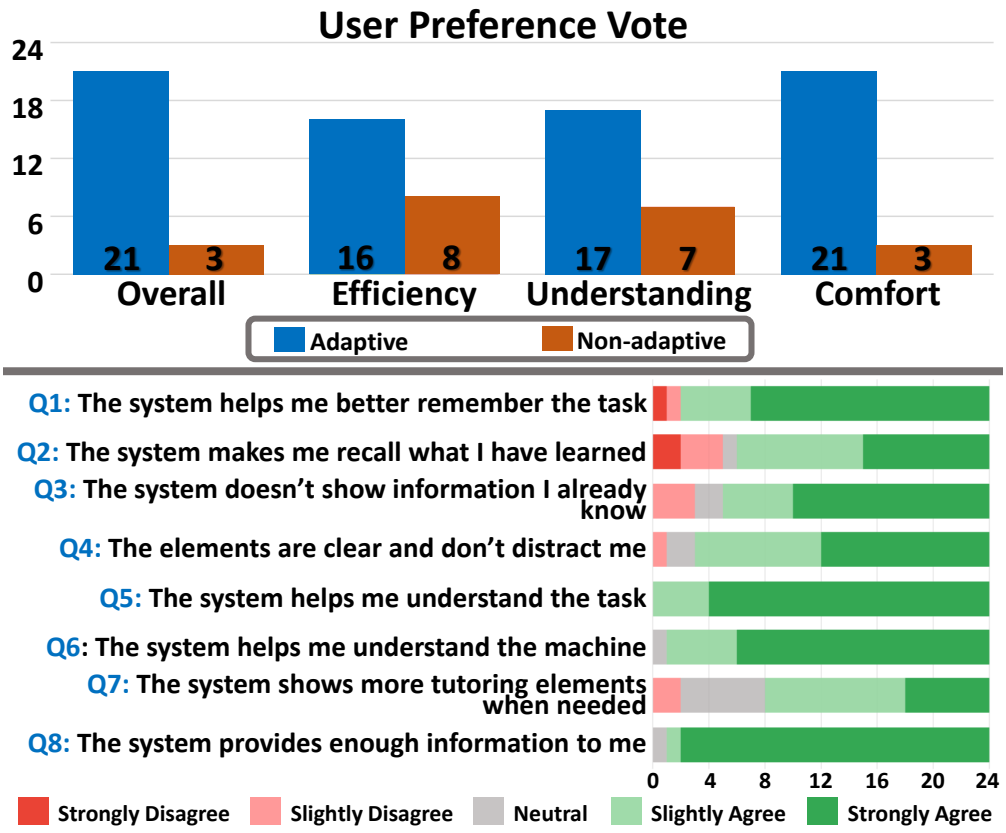

Figure 5.9. User votes and ratings for the features of the adaptive system.

By leveraging the contextual visualization of AR, the adaptation model could foster more flexible designs of AR content for future adaptive tutoring systems.

Clear view of the adaptive system. The user study results illustrate that the adaptive system can provide the exactly necessary information according to the learning progress. *"I think in most of the time, the elements showed to me are what I needed. Only when I forgot something, it showed me more." (P2)* Additionally, some users addressed another advantage of adaptively displaying the tutoring elements. *"Compared to the first (non-adaptive) system, not showing the avatar and virtual animations really increased my learning speed. Because if they were always there, then I tended to dodge them, and they really distracted me." (P8)*

Adaptive system reduces over-confidence of novice users. Subjective ratings in confidence and memorization had no significant difference between adaptive and non-adaptive system, which meant they were all confident in remembering each step (Figure 5.7). However, objective performance in testing errors showed significant difference, especially for

novice users (Figure 5.8). It revealed that novice users tended to be over-confident using non-adaptive system where all tutoring elements were always visible. *"I fully support the adaptive system because when I first used the non-adaptive one, I thought I remember everything. But I messed it up in testing. But for adaptive one, it forced me to remember and recall by hiding some elements." (P11)* Yet, proficient users clearly knew which steps needed more attention, so they had more accurate self-assessment. This expertise-dependent variation enlightens a potential research direction to developing a more sophisticated adaptation model to better assess a learner's performance.

Adaptation timing. In our system, when to display additional tutoring elements to the learners is determined by the time-sensitive adaptation model. Most users welcomed the feature where the system showed hints after they got stuck for a short period of time. However, two users mentioned that the tutoring elements sometimes appear too quickly while three users mentioned that the tutoring elements appear too late. In addition, two users mentioned that the timing of appearing tutoring elements should be more flexible. *"Maybe at the beginning, the hint can appear faster. And later, the hint appears slower for me to recall." (P18)* One potential solution is to add manual control for users to manage tutoring elements, such as using gestures (swiping their hand near the target component to uncover more details) or using voice command. Meanwhile, the system can collect the timing of manual control to fine-tune the thresholds of the adaptation model. For example, when hints are not shown, some users tend to recall without disruption, so the thresholds could be increased. In contrast, some users tend to see the hints more eagerly and may use manual control, so the thresholds could be decreased accordingly. By gradually collecting more data during the tutoring process, the system can minimize the need for users to do manual control.

The patterns of LoD change. During the user study, we logged the change of LoD for each user in the adaptive session. Referring to Figure 5.10 (top), we noticed it took three trials for the proficient users to reach level-2 LoD while four trials were taken for the novices to reach level-3. The results align with our expectation because the change of LoD is determined by the learners' real-time performance. The better a learner performs,

the quicker the LoD decreases. Such pattern of LoD variation can be used to analyze the learners' performance and also fine-tune the adaptation model for further personalization.

The reasons of LoD increment. We counted the total occurrences of LoD increment (i.e., from i to i + 1) for novice and proficient users in the adaptive session and grouped them by reasons. Referring to Figure 5.10 (bottom), we noticed that the novice users' LoDs were mostly incremented due to the unawareness of the task, while the proficient users' LoDs were mostly incremented due to interactions. This suggested that the proficient users tended to directly operate the target that they felt correct, while novice users tended to spend more time in observation. Such differences can be taken into account in developing the future adaptive systems.

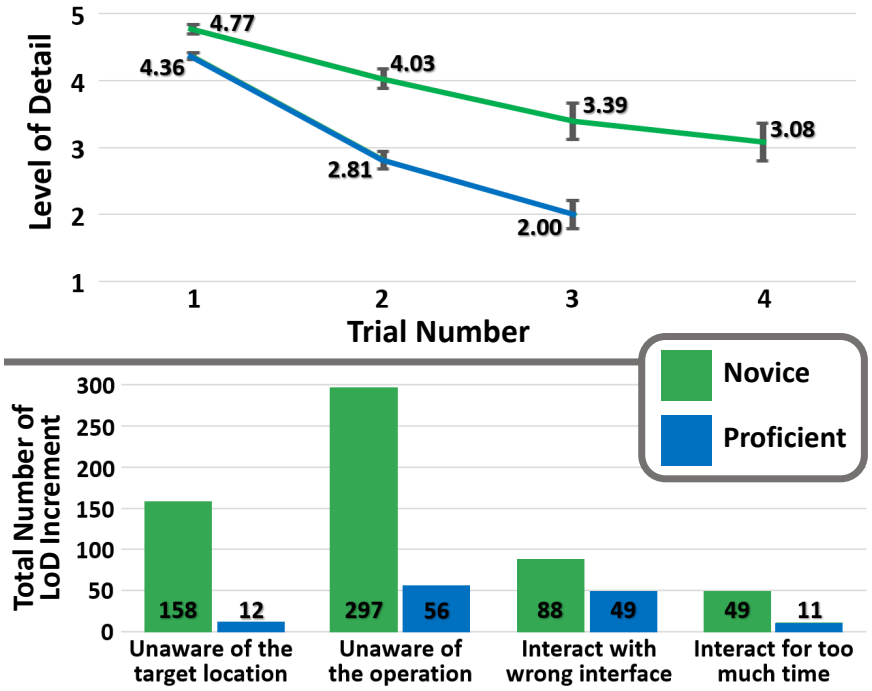

Figure 5.10. The patterns and reasons of LoD changes. (top) The average LoD of each trail for novice and proficient users in tutoring section. (bottom) The total number of LoD increment (i.e., from i to i + 1) grouped by reasons for novice and proficient users.

Evaluating AR system in VR. Under COVID-19 situation, we conducted a remote VR study to evaluate AR features of the system, which was inspired by prior works [99], [183]. Admittedly, due to the difference between the accuracy of machine state and hand

touching recognition in AR (89.1% and 93.4%) and VR (both 100%), the user study may miss some findings caused by the failure cases of recognition. However, the reported accuracy of low-level prediction in section 5.5.3 was only based on a single prediction, which was not the final accuracy to be leveraged in the high-level state prediction. We adopted a majority voting mechanism in which each prediction was decided by 5 consecutive predictions. The accuracy of machine state and hand touching recognition was increased (≈93% and ≈96%, respectively), and thus reduced the gap between the AR and VR evaluation. Moreover, since many proposed features (e.g., high-level recognition, LoD design) can be evaluated orthogonal to the low-level recognition, we can still obtain key findings from the participants (e.g., the preference of adaptive vs. non-adaptive systems, patterns of LoD change, adaptation timing, etc). In retrospect, a better approach would be to add some random failure to the low-level recognition of VR system to simulate the AR system.

Generalizability of the system. Firstly, our system supports three common types of machine tasks: local, spatial, and body-coordinated interactions [172], [173]. Many manufacturing contexts are a combination of these three types of tasks (e.g., machine tools and CNC machines) [79]. Secondly, the recognition algorithm based on images can be applied to various machines. For example, nine common types of machine components (e.g., knobs, levers) were covered in the preliminary evaluation. Thirdly, the workflow design of chaining low-level and high-level recognition can be adapted to future systems of machine tasks, rather than our system alone.

Hardware and deep learning setup. Currently, our CNN model for machine state recognition works for *discrete* states of components. We envision that more robust image based object recognition networks, Internet of Things, and hand gesture and body skeleton detection systems in the near future can provide more accurate and plentiful input information to the adaptation model.

5.8 Conclusion

In this chapter, we proposed an adaptation model that can automatically adjust the level of detail of AR tutoring elements. The model takes the input from the user and environ-

ment and performs low-level and high-level state prediction based on deep neural network and finite state machine. We also developed AdapTutAR, an AR-based adaptive tutoring system for *machine task*s that allows task authoring and tutoring via bodily demonstration. We evaluated the accuracy of the low-level state recognition on a mockup machine with 9 component types, and further evaluated the overall adaptation model via a remote user study in VR environment. In the user study, we invited 24 participants to learn tutorials using adaptive and non-adaptive systems and collected their subjective ratings and objective performance. Based on the results, we believe that AdapTutAR provides important insights for future researchers in creating an adaptive tutoring system which empowers an efficient, flexible, and productive workforce.

6. VIPO: FLEXIBLE WORKFLOWS FOR ROBOTS AND IOT MACHINES

This chapter presents the example of human-to-machine task delegation, which is delegating flexible and physical tasks to robots and machines. This chapter has adapted, updated, and rewritten content from a paper at CHI 2020 [9]. All uses of "we", "our", and "us" in this chapter refer to coauthors.

Mobile robots and IoT (Internet of Things) devices can increase productivity, but only if they can be programmed by workers who understand the domain. This is especially true in manufacturing. Visual programming in the spatial context of the operating environment can enables mental models at a familiar level of abstraction. However, spatial-visual programming is still in its infancy; existing systems lack IoT integration and fundamental constructs, such as functions, that are essential for code reuse, encapsulation, or recursive algorithms. We present Vipo , a spatial-visual programming system for robot-IoT workflows. Vipo was designed with input from managers at six factories using mobile robots. Our user study (n=22) evaluated efficiency, correctness, comprehensibility of spatial-visual programming with functions.

6.1 Motivation and Contributions

As mobile robots and human workers become more tightly integrated within IoT (Internet of Things) environments, the task of instructing the machines has become increasingly complex. With more devices to coordinate, human operators must author workflows that are inherently computational in the context of dynamic spatial environments.

Factories typify the challenges of coordinating complex workflows with mobile robots delivering parts and interoperating with manufacturing equipment. As manufacturing processes ever more increasingly dependent on customization and product changes, the effort needed to create or modify workflows becomes a bottleneck. Furthermore, some responsibility for programming robots and their interactions with IoT devices must shift to the workers directly involved with a given manufacturing process [29].

Bringing factory workers into the process will require the right level of abstraction and context. *Task-level programming* offers a starting point. In this paradigm, expert programmers write code for robots to perform generalized tasks, which non-programmers can use to direct the robot [184]–[186]. However, as basic programming skills become more pervasive, the need becomes less focused on programmers vs. non-programmers, and more on enabling domain experts to efficiently specify workflows.

We present Vipo, a *spatial-visual programming* system for robot-IoT workflows. With spatial-visual programming, programs are created using visual programming constructs drawn directly on a map of the operating environment. To start, a floor map of the physical space is uploaded into Vipo. Then, users can draw paths for robot movements, and specify loops and conditionals by connecting paths with shapes. Those constructs can also be found in other recently developed spatial-visual programming systems [119]–[121].

Vipo builds on those capabilities in two significant ways.

First, the Vipo language allows workers to write programs using *functions*. Functions are an essential building block in nearly all mainstream programming languages. Steps and data related to a meaningful sub-goal can be encapsulated in a function definition. Then, the function can be called repeatedly with different parameters. Functions can even be called recursively. Support for recursive function calls allows workers to express programs that could not be expressed without functions (or stack data structures).

Second, the Vipo architecture integrates IoT devices into the programming and execution environments with no prior configuration. Using the Vipo protocol, devices broadcast their location, capabilities, and resource status (e.g., power, supplies, etc.) in a format based on the Resource Description Framework (RDF). The Vipo architecture uses those messages to discover devices. Their status and capabilities are automatically integrated into the Vipo IDE, a web-based development environment used to create programs with Vipo. When the programmer specifies an action that involves an IoT device, its capabilities are populated into the editor, and it's resource status can be checked to ensure the actions are possible. Building on the Robot Operating System (ROS) [129], the Vipo architecture compiles the user's programs into a form that can be executed by robots, or simulated.

6.1.1 Contributions

The key contributions of this work can be summarized as follows:

- The Vipo language is a spatial-visual language for programming robot-IoT automations with functions, as well as conditions, loops, movement, and IoT device operations.
- The Vipo architecture, including the Vipo protocol, enable real-time integration of mobile robots and IoT devices with dynamic state information (e.g., locations, capabilities, resource availability, etc.).
- The Vipo IDE integrates 1) code creation, 2) testing/simulation, 3) deployment, and 4) monitoring in an integrated development environment, as a demonstration of the overarching vision for factory or other robot-IoT automation.
- Our user study with 22 participants validated the comprehensibility, correctness, and efficiency of spatial-visual programming with functions.

6.2 Design of VIPO

The key contribution of this work is the introduction *functions*—including a notation and interactions—to spatial-visual programming for robot-IoT workflows. Later in this paper, we will explain the language design and architectural challenges that were entailed to bring functions to spatial-visual programming. To establish context for that discussion, we will first explain the design goals of Vipo, and the more foundational elements of the Vipo language.

6.2.1 Requirements and design goals

The requirements and design goals were gathered through a series of visits by a group of at least three researchers to six factories. These included manufacturers of construction equipment, automobiles, electronic equipment, and components.

The visits were motivated by other collaborations related to robot-IoT automation, but on each occasion, the researchers asked questions related to the firms' challenges regarding specification of robot-IoT workflows.

Representatives also visited the lab where Vipo was developed during the development of Vipo. The researchers gave demonstrations for plant managers and executives, and received feedback that guided our understanding of the requirements.

The visits provided a series of design hints that were integral to the formation of Vipo. For example, representatives from a consulting firm that supports small- and medium-sized enterprises informed us of the cost structure of equipment acquisition, which led to key decisions related to the Vipo architecture.

6.2.2 Language design

Transitional Constructs (move)

Transitional constructs represent operations that transit from a source to a destination. They are typically used to plan the motions of mobile robots, including "move", "pick", "drop", and "carry". All four constructs can be found from the toolbar on the left of Figure 6.9. For example, "pick" means the robot picks objects from a source and moves to a destination. "Drop" means the robot moves from source and drops objects to destination. When users need the robot to pick from source and drop at destination, they can use "carry", which is a combination of "pick" and "drop". Transitional constructs can represent the spatial relationship between devices, such as the direction and distance. Figure 6.1 shows four transitional constructs between the paint inventory to paint mixer.

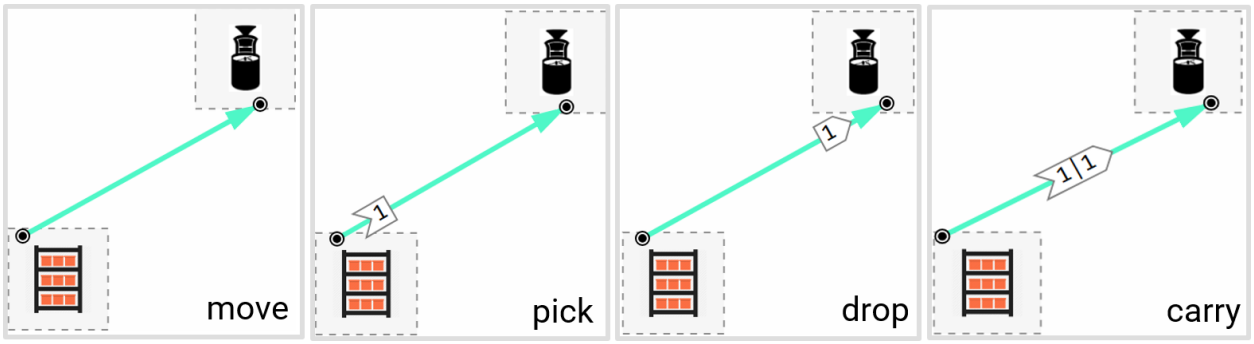

Figure 6.1. Examples of four transitional constructs.

There are two reasons why we design distinct notations for "pick", "drop", and "carry", rather than reusing "carry" as a universal notation. Firstly, notations can leverage spatial

proximity. Specifically, "pick" is spatially closer to the source, "drop" is closer to the destination, and "carry" sits in the middle, as shown in Figure 6.1. Secondly, we want to use the distinct shape to enhance the readability and memorability. If users get familiar with the notations, they can immediately tell if a notation is a "pick", "drop", or "carry". To strengthen the concept that "carry" is a combination of "pick" and "drop", the symbol of "carry" is also a combination of the symbols of "pick" and "drop".

In-place constructs (IoT operations)

In-place construct is used to plan an operation for an IoT machine. When the robot carries materials to a machine, the machine has various ways to interact with the materials, such as consuming, processing, and/or packing. A "Timer" symbol is used to reveal an estimated execution time of this operation. If users click on the Timer symbol, a popover will appear to allow users to choose a capability of the particular IoT and specify how long it will take and what extra parameters are desired. Figure 6.2 shows an example in which the paint mixer is programmed to mix paint for about 5 minutes.

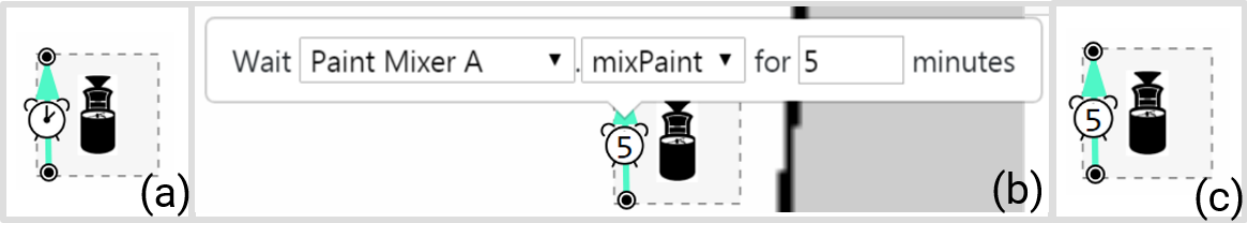

Figure 6.2. Example of in-place construct: mix paint for 5 minutes. (a) start to create a timer, (b) use popover to select a machine capability and enter required parameters, (c) show estimated time in timer.

Control-flow constructs (if and loops)

The control-flow constructs include "if" and "while". Both "if" and "while" use a condition to determine which of two paths to follow. A condition can include properties of devices, operators, and/or numerical values. Properties of devices are typically the sensory data of devices, such as the working status of a printer, the current temperature of an oven, or the

empty slots of a shelf. Figure 5 shows an example in which the robot is asked to drop the paint can at mixer A if its jobStatus (i.e., completion progress) is greater than 80 percent, otherwise drop at mixer B.

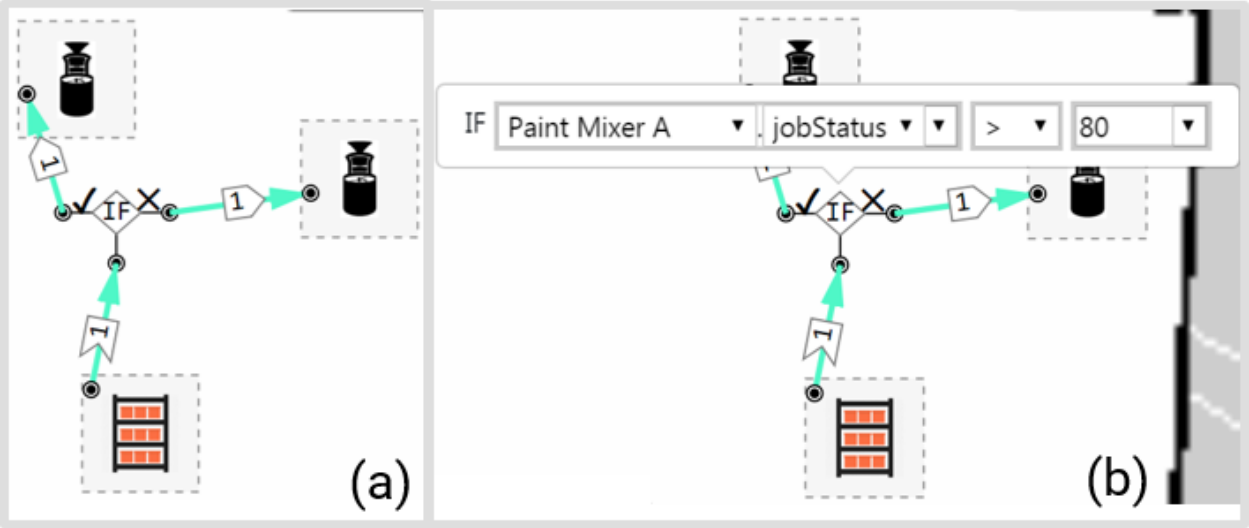

Figure 6.3. Example of "if". (a) The if condition decides which of two branches to follow, (b) Users select a property of a device, a logic operator, and enter a value to specify the if condition.

Besides "if" and "while", the aforementioned transitional constructs could also be considered as one kind of control flow constructs: "goto". In most cases, a workflow of a robot is a linear sequence of actions. However, sometimes users may draw a path (e.g., one of transitional constructs) that goes back to its prior action, which forms a loop. When this backward "goto" path is combined with "if", the workflow is functionally equivalent to a "while" loop.

6.2.3 Spatial Functions

This section shows how the Vipo language supports function definition and function call in the spatial domain.

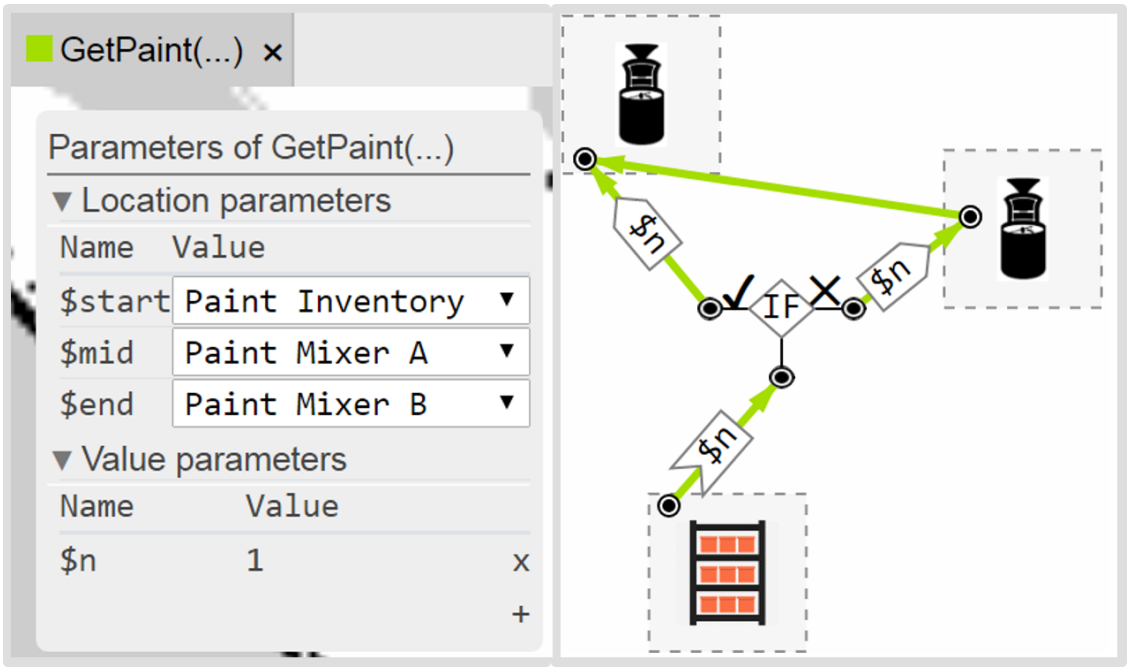

Figure 6.4. Function definition. Define a value parameter "$\$n$" for function "GetPaint" (left); use "$\$n$" in pick, drop, and if-condition (right).

Function definition

A function is used to represent a workflow that can be tested and executed alone, or reused by other functions via function call. A function can include one or more primitive constructs, or even functions. Each function has a distinct color to differentiate from other functions. The constructs belonging to a function have the same color as the function. To support people with color blindness, users can use different color value.

Functions are displayed as a list on the top-right panel of the editor (Figure 6.9). The user can click on a function to display its content in the main workspace within a tab. A click on another function opens a new tab. All functions for the same environment share the same layout map and machines, and are organized by tabs.

Similar to textual programming languages and other non-spatial visual languages, defining functions with parameters can make the workflow more flexible and customizable. There are two types of parameters: value and location; both of them start with a dollar sign "$".

- Value parameter (e.g., $\$n$). A value parameter is used to store a number. With the help of value parameter, users may replace a constant number with an algebraic expression. An algebraic expression can be a constant number, a variable, or algebraic operation on algebraic expressions (e.g., $\$n \times 2 + 3$). For example, the number of objects to pick/drop/carry can be "$\$n$" instead of a constant number (Figure 6.4). Likewise, the condition of "if" and "while" can use an algebraic expression.

- Location parameter (e.g., $\$start$, $\$end$). A location parameter is used to store a machine. With the help of location parameter, users may change some actions that happen on one machine to happen on another machine. For example, a robot may visit and interact with machine A, machine B, and machine C in linear order. Rather than visiting a sequence of fixed machines, users may convert machine B as a location parameter (e.g., $\$middle$). In such case, if we pass machine D into $\$middle$, the robot will eventually visit and interact with machine A, D, and C.

Function call

A function can reuse existing workflows via function call. If users want to call a function, they can right-click the desired function in the function list and select "Call this function" from the context menu. Then users can click on the map to specify the start and end of the function. This drawing operation is the same as transitional constructs. The visual notation of a callee (i.e., the function being called) also looks like a transitional construct, as shown in Figure 6.5a. The visual notation includes the function name and an "Expand" button. Once clicking on the "Expand" button (Figure 6.5b), the internal definition of the callee will be displayed (Figure 6.5c). This allows users to quickly peek the definition without switching between functions. The path color of a callee remains the same as the callee, which makes it distinct from the constructs belonging to the caller (i.e., the function that calls callee).

To further customize the workflow in callee, users can pass different values and locations to its parameters. If users click on the callee notation, a popover will appear to allow users pass a different values to the callee. Similarly, users can select a different device for a location parameter. Once a location parameter is assigned to a new device, the workflow

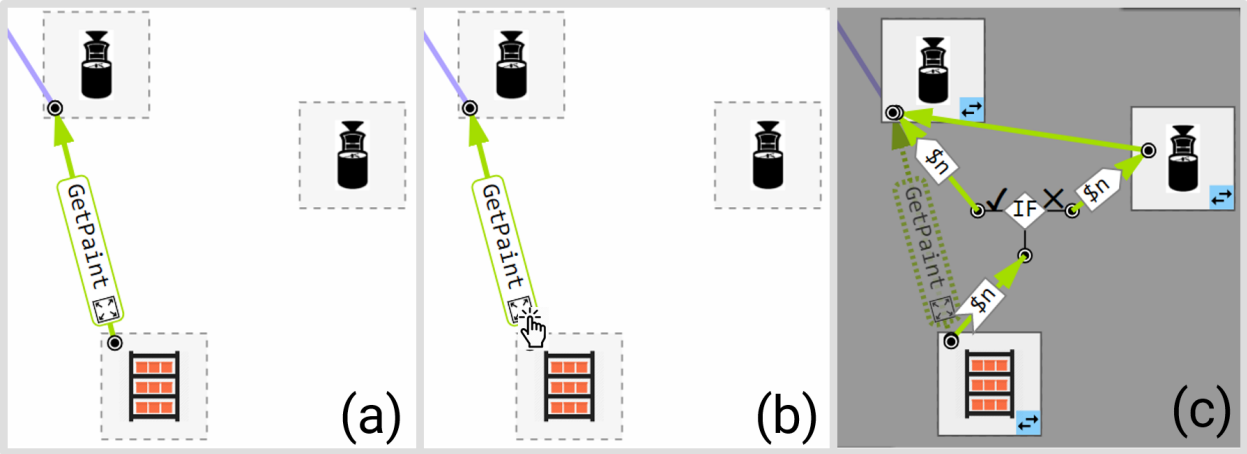

Figure 6.5. Function call and assign values. (a) call function "GetPaint", (b) click to expand the detail, (c) the detail of "GetPaint" is displayed within lightbox

to be executed is changed to visit and interact with the new device. This change can also be visualized if users click the "Expand" button to see the expanded detail of callee. Note that this change will only affect the execution of callee within the current caller, while the original definition of the callee function is not affected.

In addition, since users can expand a callee to see its internal detail, it enables a unique way of assigning location parameter. Users can click the "SwitchDevice" button near the bottom-right corner of each device. Then users can click on a different device and assign the new device to the corresponding location parameter, as shown in Figure 6.6. The originality of our approach is that a location parameter can be spatially visualized and also can be assigned to a new device spatially. Once the location parameter is successfully updated, the constructs that are related to this location parameter will be automatically switched to the new device.

6.3 Architecture

The system can be treated as a three-layer architecture, as shown in the Figure 6.7. From top to bottom, it includes a task planning layer (i.e., Vipo), a task control layer (i.e., ROS Master), and a task execution layer (e.g., robots and IoT devices).

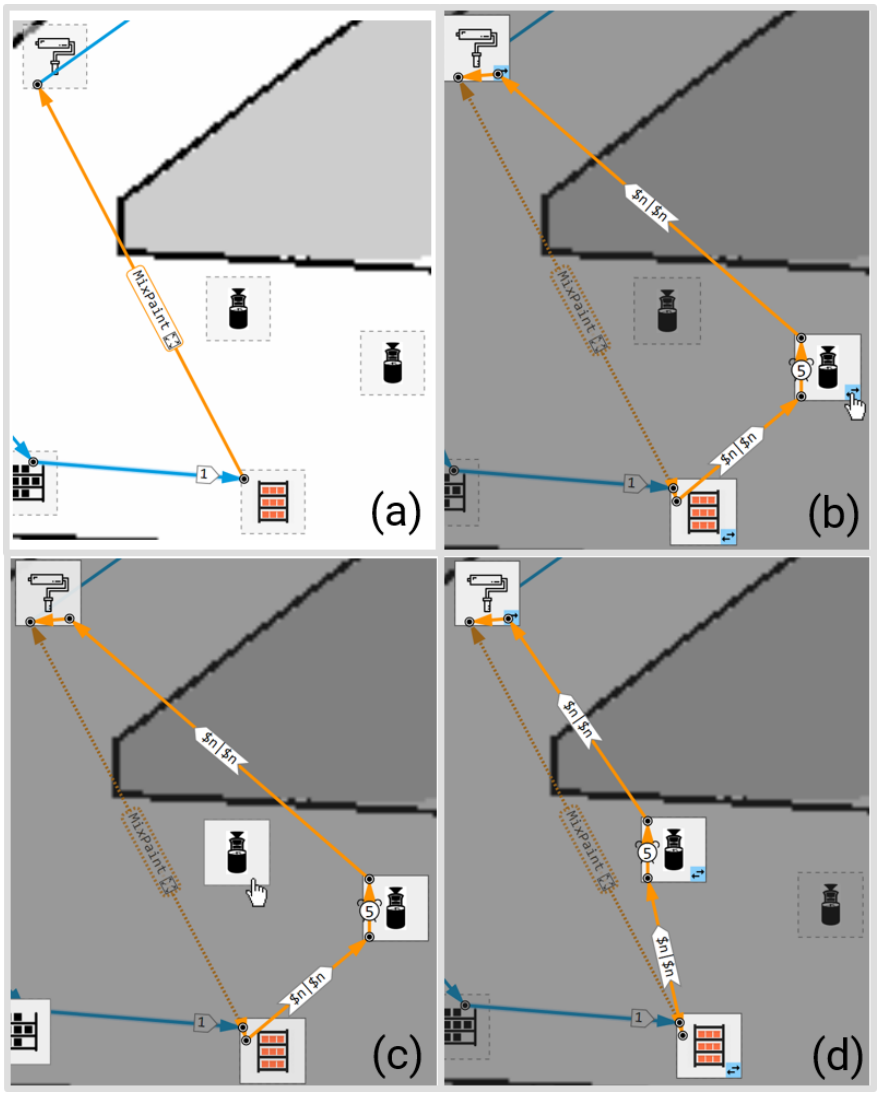

Figure 6.6. Assigning a new location in expanded callee view. a) The callee "MixPaint" before expanding, b) click on the "SwitchDevice" button in the expanded view of callee, c) click on a new device, d) location parameter is successfully changed and the constructs are switched to the new device automatically.

The task planning layer is part of the Vipo IDE, a web-based development and simulation environment that allows users to program tasks for robots/IoT devices. More details are given in later section.

The task control layer is ROS Master which acts as the bridge of the two-way communication between Vipo and the robots/IoT devices. When RDF message is sent from

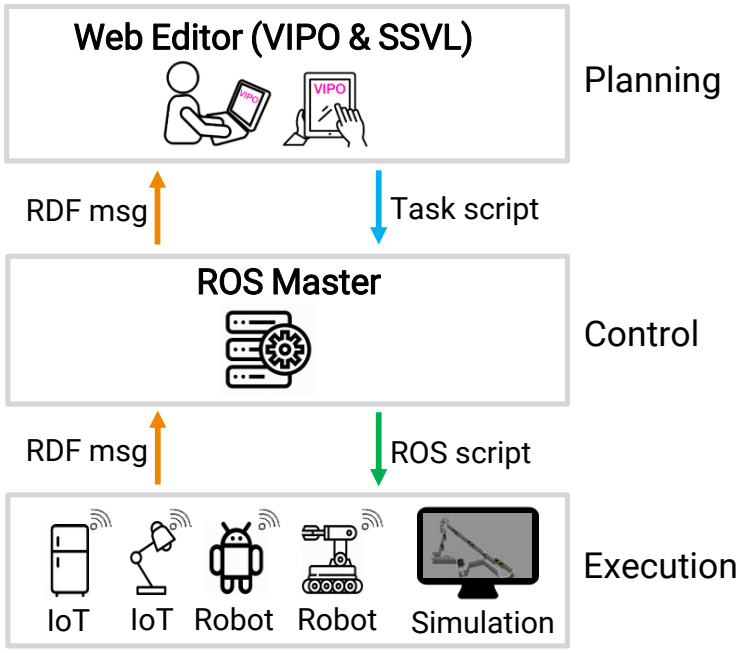

Figure 6.7. The three-layer architecture.

robots/IoT, ROS Master maintains a global context that keeps track of all the connected devices. ROS Master only sends the difference of two adjacent RDF messages from the same device to Vipo to reduce network traffic.

When a task script is sent from Vipo, ROS Master creates a thread for each new task. The task script receives the global context to fetch the details of the corresponding device. Each line of task script is translated into ROS-specific code and send to a corresponding machine.

The task execution layer consists of physical or simulated devices (robots/IoT devices). Each device holds the spatial information (e.g., location), sensory data (e.g., temperature and job status), and functionalities (e.g., packing a box, 3D printing).

IoT devices and robots are periodically publishing RDF message to ROS Master while subscribing command script from ROS Master.

In our system, we adopted a modified version of RDF to suit our application. Figure 6.8 presents a sample RDF message for a paint mixing machine. The modified RDF message has device-information fields (ID, name, description, location etc.), machine-specific *methods* and *properties*. *Methods* are the functions that the machine is capable of performing (like

mix-paint, set-temperature, start, stop etc.) while *properties* are the real-time operation parameters (like job-status, temperature, coolant-level, run-time, health-status etc.)

The information from the RDF Message serves some features of Vipo. The RDF message fields discussed above are directly used by Vipo for different purposes. Specifically, fields like the location, iconUrl, and iconSize are used for rendering the icons on the map. The *properties* fields are used for populating the drop-downs of the control-flow constructs (e.g., jobStatus used in Figure 5.). Moreover, the *methods* fields are used to populate the drop-downs of in-situ operation constructs as callable functions.

Given that the capabilities of IoT devices and robots often need users to specify some values, the *methods* fields of RDF message can automatically provide the parameter interface for capabilities. Vipo uses the *vipo_msg_type* field as the parameter interface that specifies what kind of value the method requires. For example, the "mix-paint" method of a paint mixer requires users to specify a time. Similarly, a "move" method of a robot needs to specify the location to move. So far we have supported four types: time, object, location, and value, which can be easily extended to support more types.

6.3.1 Communication Between Layers

Each layer keeps sending and receiving message from the adjacent layer(s) (*S2*). Overall, it includes bottom-up and top-down communication.

Bottom-up: Broadcasting spatial and contextual information. The goal of this communication is to broadcast the spatial and contextual information from the execution layer to the planning layer. The broadcasted messages are used to setup the programming environment and reflect the real-time status of robots/IoT devices.

Specifically, each device is periodically broadcasting its spatial location, sensory data and functionalities to the ROS Master. The format of the message is based on a modified version of Resource Description Framework (RDF), which will be described in later section.

The ROS Master forwards these RDF messages to Vipo, which further renders each robot/IoT at the corresponding location defined in RDF messages. Moreover, the function-

```
------------------
Property.msg
   string name
   float32 value

------------------
Method.msg
   string name
   string vipo_msg_type
   string topic_name

------------------
RdfMsg.msg
  string id
  string name
  string description
  float32[] location
  string size
  string imgUrl
  bool done
  bool error
  Property[] properties
  Method[] methods
```

```
---
id: "paintMixer02",
name: "Paint Mixer B",
description: "Mixes paint for a
              given duration of time",
location: [14.7755,-5.8476,0.5699],
size: "medium",
imgUrl: "/imgs/mixer.png",
done: false,
error: false,
properties:
 -
  name: "jobStatus",
  value: 50
methods:
 -
  name: "dispense",
  vipo_msg_type: "object",
  topic_name: "/dev05_dispense"
 -
  name: "mixPaint",
  vipo_msg_type: "time",
  topic_name: "/dev05_mixPaint"
---
```

Figure 6.8. The schema of modified RDF (left) and a sample RDF message (right).

alities in RDF messages are converted to callable functions that can be used to program a task.

Top-Down: Deploying Task. The goal of this communication is to deploy the task(s) programmed by workers from the planning layer to the execution layer. When workers deploy a task, the Vipo architecture first compiles the visual program into a textual task script, which sends to ROS Master. ROS Master receives the task script, interprets and executes it line by line. It converts each line of script into ROS-specific command and send to a corresponding robot or IoT device. Finally, robots/IoT devices receive the command and execute. The execution status (e.g., success, error) is sent back to ROS Master as an RDF message.

6.3.2 Implementation

The task planning layer (Vipo IDE) is a web-based application. The server side is using Node.js[1] (version 10+) and Express.js (version 4), with 1k sloc. The client side is built on React.js Framework[2] (version 16.5) in TypeScript language (13k sloc). We use Socket.io[3] to achieve two-way communication between the server and client of Vipo IDE.

The task control layer is written in Python (1k sloc). To make use of ROS, the library called *rospy*[4] is used. We use Socket.io to achieve two-way communication between the server of Vipo IDE and ROS Master. The task execution layer is described in the Factory Use Case section.

6.4 Vipo IDE - Task Planning Layer

Vipo IDE has three work modes: Edit, Test, and Deploy, as shown in the top-right corner of Figure 6.9. In Edit mode, users can program workflows in the Vipo language. In Test mode, users can simulate what the workflow would do before being deployed to a physical environment. In Deploy mode, the visual programs are compiled and sent to robots/IoT devices for execution. At the same time, the real-time status of robots/IoT devices is monitored and viewed in the editor. This section introduces how to setup the interface for a new environment, and then introduces three work modes of the Vipo IDE.

6.4.1 Setup

Vipo receives the information from the execution layer to set up the environment.

The layout map as the background canvas

At the start of the application, the map of the environment is displayed, as shown in Figure 6.9. The map is generated by the robot after scanning the environment (e.g., via

[1]↑https://nodejs.org/en/
[2]↑https://reactjs.org/
[3]↑https://socket.io/
[4]↑http://wiki.ros.org/rospy

Figure 6.9. The Vipo IDE displays a toolbar (left), three modes (top-right), and a 2D layout map with IoT machines at the corresponding location (center).

LIDAR). At the same time, while the robot is scanning, the locations of IoT devices are obtained (similar to [187]). Both the map and the locations are sent to ROS Master and eventually to Vipo, as described in the architecture.

IoT Device Registration

Similar to the map generation, IoT devices keep sending their contextual information (e.g., id, name, status, and supported capabilities) to ROS Master then to Vipo. Thereafter, Vipo automatically renders the IoT devices as icons at the corresponding locations on the 2D layout map, as shown in Figure 6.9. The icon image is also defined by the IoT itself.

6.4.2 Edit Mode – Program with The Vipo language

Based on the layout map and icons of IoT devices in the environment, we designed the Vipo language to program workflows. The basic statements of the Vipo language are spatially oriented. We have incrementally introduced the syntax of the Vipo language and showed how to program in the earlier section.

6.4.3 Test Mode – Simulate Execution

Testing is a mandatory activity before deployment. Users can click the "Test" button to enter test mode and simulate how the workflow will be executed by a robot. Three control buttons are provided, including playing all steps at once, playing one step at a time, and starting over from the beginning (Figure 6.10a). A robot moves along the path with animation. If there is a control-flow constructs, the condition is evaluated to choose which branch to follow (Figure 6.10c). The condition may involve the properties of machines, which are dynamic and external. To enable testing, users can enter test value to mock the properties (Figure 6.10b). By passing different test value, the robot is able to move along both branches of an "If" statement so that the test coverage is more comprehensive.

Before the robot moves to the next construct, Vipo checks the syntax of the next construct and evaluates its value. For example, if the number to pick/drop is missing or the condition of "If" statement is not completely filled, a red bulb icon will be displayed with error message (Figure 6.9). Moreover, if the algebraic expression has wrong syntax or the variable is not defined, the red bulb icon will also be shown to give the error message. The robot pauses movement until users fix the error.

Unlike other programming languages where the programmed script and the simulation result are visualized in separate interfaces, the Vipo IDE can show the programmed constructs and simulation result in the same interface. In other words, the simulation is running directly on top of the programming constructs within the environment. This direct coupling may enhance the predictability of the simulation result. In a loose sense, Vipo supports *"What-You-See-Is-What-You-Expect-To-Get"* in the environment.

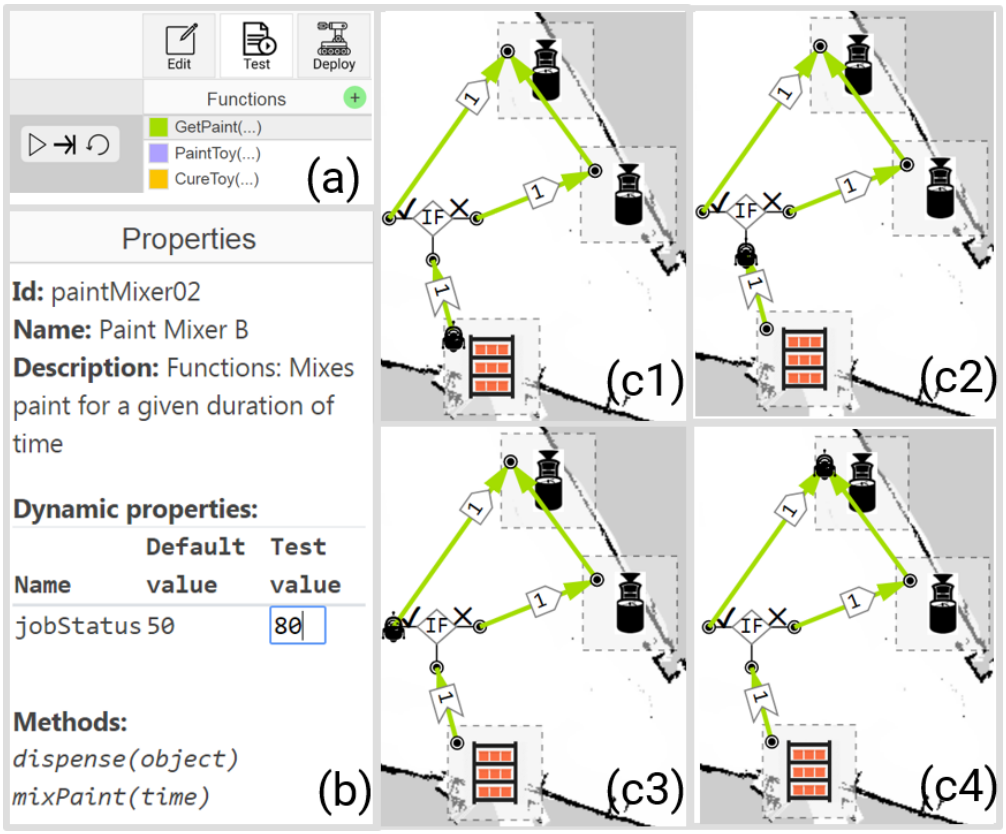

Figure 6.10. Test mode. a) Users switch to the Test mode and use three buttons to control the simulation, b) users set test values for dynamic properties of devices to simulate different execution results, c) once users click the play button, a robot moves along the path and chooses the proper branch to follow based on the if condition.

6.4.4 Deploy Mode - Execute and Monitor Status

The Vipo IDE can monitor the real-time execution status of robots and machines, and allows users to correlate the robot's movement with a particular visual construct. If the physical robot is moving, it is shown in the interface at the corresponding location. Since the visual constructs of the task is also displayed in the interface, users are able to recognize which construct the robot is currently executing. This direct mapping between the execution in physical environment and the programming constructs in digital layout can help users better understand the current state and predict the next state.

Moreover, if a machine reports an error during execution, the Vipo IDE shows a red mark to highlight that machine. This real-time error reporting allows users to notice the error quickly and fix it to increase productivity.

The reason to have Edit mode and Deploy mode is that it would be distracting to see a moving robot while programming. By switching back to the Edit mode, the RDF messages at that time are cached and used to render the machines statically in Edit mode. The subsequent RDF messages are ignored until switching back to deploy mode.

6.5 Use Cases

To show how Vipo can be used to program tasks for robots/IoT devices, we present and explain two use cases of our system.

6.5.1 Scalability and Reusability - Recursive Function

The first use case demonstrates that the Tower of Hanoi puzzle[5] can be expressed in the Vipo language. This puzzle is often used as an example to teach recursion in computer science courses. Figure 6.11 shows a recursive solution expressed in the Vipo language.

To solve this puzzle, a robot needs to pick, move, and drop a disk, which fits well with the capabilities of a mobile robot. In addition, the three rods are spatially situated, which can be implemented as IoT devices. In other words, the Tower of Hanoi puzzle represents a simple but non-trivial task for robots and IoT devices.

The ability to define parameters and use function calls enables users to program more complex workflow in a spatial domain, including recursive functions.

6.5.2 Factory Use Case

To validate the system developed in an industrial context, we deployed Vipo to author robot-IoT workflows for simple material-handling applications in a small-scale emulated painting factory. The industry considered is assumed to have an advanced factory

[5]↑https://en.wikipedia.org/wiki/Tower_of_Hanoi

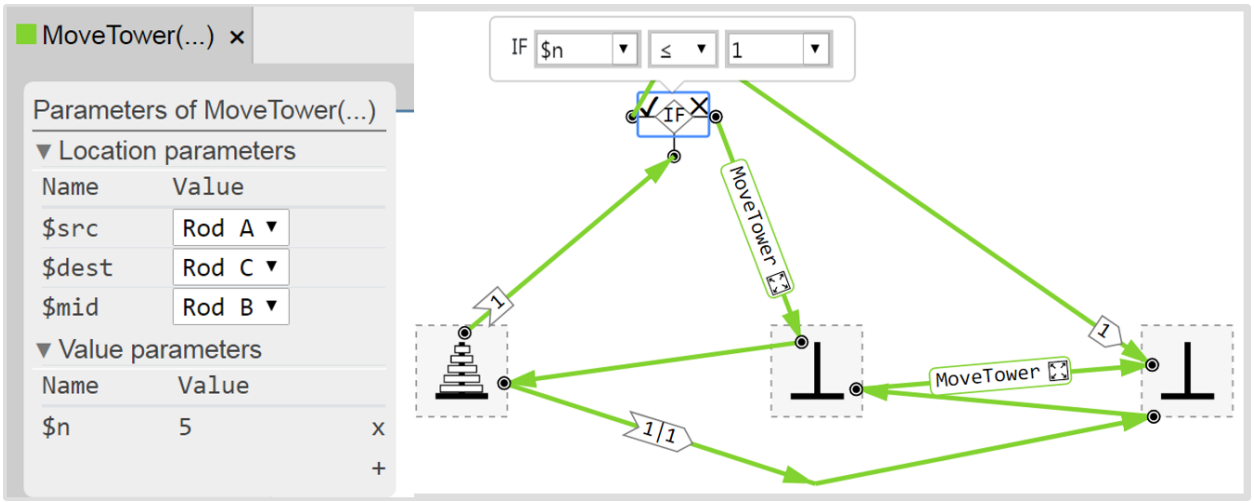

Figure 6.11. Recursive function calls enables a recursive solution to the classic Towers of Hanoi.

setup with smart machinery and autonomous mobile robot (AMR). Workflow of a typical job order comprised of the following sequential steps:

1) Source the parts to be painted from the Item Inventory
2) Source paint cans from the Paint Inventory
3) Mix the paint to obtain a given shade using Paint Mixer
4) Paint the parts using the Painting Machine
5) Cure the painted parts at a given temperature in the Curing Oven

System setup (Figure 6.12) for the use case comprised of (a) an omnidirectional robot capable of autonomous navigation using LIDAR (SICK TiM561) and SLAM, (b) a 6-DOF robotic arm mounted on the mobile base for pick and drop operations, (c) Six ESP32 microcontrollers emulating six smart industrial machines, (d) ROS-Master for task flow and execution, and (e) the Vipo IDE for programming the workflow.

Prior to physical deployment, the workflow was programmed and simulated in edit and test modes of Vipo. Then the program was deployed to the robot to complete the programmed sequence in the emulated painting factory.

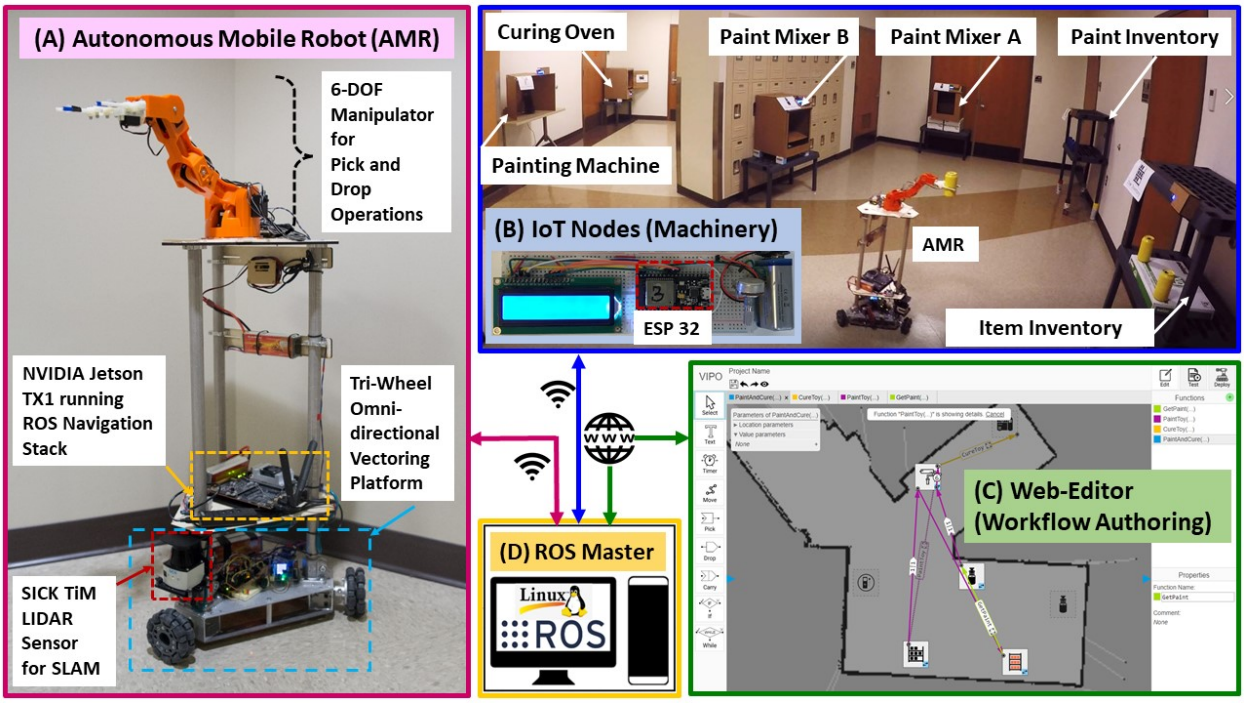

Figure 6.12. System Setup for the factory use-case (A) Autonomous Mobile Robot (B) IoT Nodes (industrial machinery) (C) The Vipo IDE (D) ROS Master

6.6 User Study #1: VIPO vs. Blockly

To understand the strength and limitation of spatial-visual language, we conducted a user study that compared Vipo with a non-spatial visual programming tool called Blockly [118]. Here, "spatial" means the programming constructs that contain spatial information, such as direction and distance.

Blockly is chosen as the non-spatial baseline due to three reasons. First, at the visual programming language level, the Vipo language and the block-based visual language of Blockly are imperative programming language and thus able to specify task for robots as a set of actions. This is unlike other dataflow-based or event-triggered visual languages, such as Node-RED [102]. Second, Blockly can create customized blocks to represent machines, functions, and properties, which can be used to program the same tasks as Vipo. Third, Blocky is a well-established visual programming tool and has been widely adapted for pro-

gramming educational purposes, which serves as a solid baseline for evaluating Vipo in terms of usability.

In such setting, the main difference between Vipo and Blockly is that Vipo uses spatial constructs directly on a 2D map, while Blockly uses non-spatial constructs with a 2D map in a separate view.

6.6.1 Experiment

Twelve participants were recruited for the study, of which most are engineering students between the ages of 19-22 years. Of the 12 total participants, 11 participants were novice programmers (0-1 year of experience), and only 1 participant was an experienced programmer (3+ years of experience). 3 users had previous experience of visual programming in Scratch (¡ 6 months of usage).

A within-subject study was conducted in which each participant was asked to use both Vipo and Blockly to program tasks in counterbalanced order. In other words, six participants used Vipo first and then Blockly, while the other six participants used Blockly first and then Vipo.

For each interface, participants had to watch a video tutorial, finish six tasks (Task 1-Task 6), and finally fill in a questionnaire regarding the user experience and cognitive dimensions for interface evaluation [188]. In the first five tasks, participants were asked to program workflows using basic programming constructs (e.g., move, pick, drop, and If-Else), as well as more advanced programming constructs (e.g., Function calls). The first two tasks (Task 1 and Task 2) were designed such that each was a standalone workflow (e.g., "PickAndPack" and "StoreItem"), but also could be reused in Tasks 3-5 to form a more complete workflow via function calls. Participants were allowed to use the Test-Mode for assistance (debugging). Once participants finished all five tasks, they were asked to read and comprehend a task programmed by the authors (Task 6). While participants were programming using the interface and filling in the questionnaire, the computer screen was recorded.

Table 6.1. Results of usability test in 9 cognitive dimensions. None of them have significant difference.

	Blockly		**Vipo**	
	Mean	SD	Mean	SD
Visibility	4.08	0.67	4.08	1.08
Viscosity	4.58	0.67	3.92	1.38
Diffuseness	3.91	0.79	4.42	0.90
Hard-mental operations	3.91	1.08	4.50	0.90
Error-proneness	3.41	1.38	3.50	1.31
Closeness of mapping	4.00	0.47	4.17	1.03
Role-expressiveness	2.25	1.29	2.42	1.51
Progressive evaluation	4.33	0.78	4.42	0.79
Premature commiment	3.75	1.06	3.25	1.60

The first five tasks are the same for both conditions, which are used to compare the system under the same context. The last reading task is to test comprehension. However, in order to prevent participants from repeating their answers from the previous interface, we kept the bulk of the programs the same while used slightly different numbers in If-Else condition. The hypothesis of this experiment is spatial-visual programs created in the Vipo language are more comprehensible than functionally equivalent programs written in a non-spatial visual programming language.

6.6.2 Results & Discussion

In this section, we first report the results in comprehension test, then report usability test results.

Spatial constructs make programs more comprehensible

In the comprehension test (Task 6), participants were given a programmed workflow and asked to answer five questions, such as identifying the optimal path, number of machines involved, and the source and destination of the program based on different if-condition. Each question counts as 1 point. We use the total score of the five questions to represent

a participant's understanding of the program. A paired t-test was conducted to compare the comprehension test results in Vipo and Blockly. There was a significant difference in the scores for Vipo (M=3.83, SD=1.03) and Blockly (M=2.83, SD=1.53), t(10)=2.57, $p = 0.03 < 0.05$. The results suggest that participants had a better understanding of the program when the workflow is programmed in the Vipo language, which supports our hypothesis that spatial visual programming language improves user's comprehension of the program. The reason might be that the Vipo language shows the programmed constructs in the same interface, therefore participants can easily infer the results of the workflow without any context switching. On the contrary, participants have to constantly switch between the map and the constructs to understand the execution result of the workflow in Blockly, which increases the chance of making mistakes.

Usability of Vipo is on par with Blockly

Next, we look at the cognitive usability test results reported by the participants. After completing all of the programming tasks (Tasks 1-5), participants were asked to evaluate the system by answering questions in 9 different cognitive dimensions [188]. Results are summarized in Table 6.1. No significant differences were found between Blockly and Vipo in each dimension (with all $p > 0.05$), which indicates that both interfaces resulted in similar user experiences.

Correctness and Time spent

Finally, we acknowledge the time spent and correctness in both interfaces. The time spent is defined as the total time spent on finishing Tasks 1-5. Correctness is defined as the number of tasks that were done correctly by the participants in all five tasks. Two paired t-tests were conducted respectively to see whether there were significant differences in time spent and correctness. No significant differences were found in the correctness for Vipo (M=3.25, SD=1.29) and Blockly (M=3.08, SD=1.44); t(10)=0.4, $p = 0.7$. There was a significant difference in the total time spent (seconds) for Vipo (M=1748, SD=559)

and Blockly (M=2566, SD=1075); t(10)=0.4, $p = 0.009 < 0.05$. The results suggest that participants spent less time in completing all five tasks with Vipo.

6.7 User Study #2: Function vs. Non-function

In this study, we investigated the pros and cons of supporting *functions* in the spatial domain. Participants were asked to program tasks in two conditions: using Vipo with functions (**condition A**) and without functions (**condition B**). The following hypotheses are to be tested.

6.7.1 Experiment

Ten participants were recruited for the study, consisting of mostly engineering students between the ages of 20-31 years old. Of the 10 total participants, 6 participants were novice programmers (0-1 year of experience), 2 participants were beginners (1-3 years of experience), and 2 participants were experienced programmers (3+ years of experience). 2 users had previous experience of visual programming in LabView (¡ 6 months of usage).

Participants started with a video tutorial that explained all features of the Vipo language except functions, and then completed three tasks (Task1-Task3) as warm-ups. The authors verified the accuracy of warm-up tasks and explained any issues that occurred. Next, each participant was asked to use both condition A and condition B to program tasks in counterbalanced order. They had to complete three tasks in each condition (Tasks 4a-6a for condition A and Tasks 4b-6b for condition B), then fill in a questionnaire, and proceed to the other condition. For condition A, participants needed to watch a second video tutorial that included functions. Once they have done both conditions, an exit questionnaire was given to compare both. While participants were programming, the computer screen was recorded.

Tasks in both conditions were the same, except that tasks in condition A required the use of function calls while tasks in condition B required the use of basic notations directly. For example, in Task 5, participants were told that they act as a maintenance worker trying to fix a broken machine. Therefore, they need to program robots to collect tools as well as

three types of replacement parts from inventories, use a cutting machine to cut to specific shapes, and then carry to the broken machine. In condition A, participants can define a function called "getPart" and call it three times with different numbers and locations.

6.7.2 Results & Discussion

In this section, we first report the results from the questionnaire, then report efficiency and accuracy based on screen recording and created programs.

Functions had less viscosity

The questionnaire asked participants to answer questions in the 9 cognitive dimensions. A paired t-test was conducted for each dimension. We did not find a significant difference in eight dimensions. However, there was a significant difference in viscosity between condition A (M=4.50, SD=0.53) and condition B (M=3.70, SD=1.16), $p = 0.04 < 0.05$. The reason might be that participants in condition A only need to update inside the function being called, while participants in condition B need to update all occurrences.

Functions made programming faster

The total time of completing Tasks 4-6 was measured. A paired t-test was conducted to compare the total time. No significant difference was found between condition A (M=17.1) and condition B (M=20.9), t(9)=-2, $p = 0.1$. However, if we look at Task 5 where the participants were asked to program robots to do the same operation three times, the usage of function saves time for doing the same operation. As Figure 6.13 shows, participants in condition A spent about one and a half times longer on the first function call (2.63), compared to condition B (1.96). However, for the remaining two sub-tasks, condition A took about half of the time as that of condition B. This shows that the learning curve of function is higher at first but it helps participants more efficient once mastered. Participants in condition A can reuse the workflow while those in condition B have to repeat the same steps. The time in condition A of making a function call is constant while the time in condition B is proportional to the number of steps.

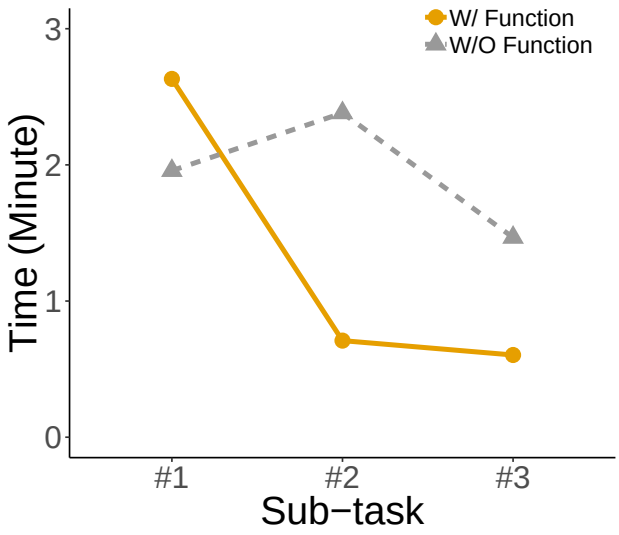

Figure 6.13. Learning to use function takes time, but the efforts pay off when the same operation is being used several times.

Functions were more error-prone

The accuracy of the programs were measured. A paired t-test was conducted to compare the number of correct tasks in two conditions. There was a significant difference in the accuracy between condition A (M=2.20, SD=0.63) and condition B (M=2.70, SD=0.48), t(9)=-2, $p = 0.05$. The result suggests that participants made more errors when using functions, compared to using just basic constructs. This is surprising but understandable. Participants need to learn how to accurately use function in a short time which includes: defining a parameter for the function, replacing the constant number in the function with the parameter, and finally passing a different value to the parameter. These extra steps were challenging for novice programmers and increased the error proneness. After checking the created programs, we found one bug that was particularly common and resulted in the significant difference: five participants in condition A forgot to replace the constant number in the function with the defined parameter. This suggests that we need to improve the editor by highlighting unused variables.

6.8 Limitations and Future Work

The system is designed to program tasks for one robot. In fact, a task may need different types of robots or a collaboration of multiple robots. Such complexity is abstracted away from the task planning layer (Vipo), but a more intelligent ROS Master is required to manage the execution. Furthermore, we assume a robot is executing one task at a time. In a real factory, the robot may be shared among different tasks, in which efficient scheduling and optimization algorithms are required [189].

Vipo was not tested with workers within real factories, because some issues should be addressed first, such as adapting existing machines to use RDF messages, and handling exceptions during execution.

The current interface has limited visualization on the real-time status of machines (only the location and success/failure). In the future, it would be more informative to allow users to customize the visualization of more kinds of status.

In the future, we envision bringing humans into the workflow more explicitly. Workers would broadcast their location and other available status information via RDF messages. In such case, users will be able to program tasks to control the collaboration between humans, mobile robots, and machines.

6.9 Conclusion

Vipo supports modular visual programming of robot-IoT workflows in the spatial context of the operating environment. Using the Vipo IDE, users can create, test/simulate, and deploy/monitor automations visually, spatially, and interactively. We implemented two use cases to demonstrate that 1) the Vipo language can support complex programs involving recursive functions and 2) the tasks programmed in Vipo can be executed by robots and machines in physical environments. The user study with 22 participants indicates that 1) spatial constructs make programs more comprehensible, 2) functions can make programming faster.

7. OVERALL CONCLUSION

In this dissertation, I have presented four systems that tackle the challenges in human-to-human and human-to-machine task delegation.

On one hand, BlueSky (in Chapter 3) and CoStory (in Chapter 4) are two systems that aim to delegate creative work to human workers. To leverage the diversity of large group of human workers, these two systems first identify the characteristics of crowdsourcing and friendsourcing (e.g., low level of commitment), and then design workflows that mitigate their relatively low level of determinism.

On the other hand, AdapTutAR (in Chapter 5) and Vipo (in Chapter 6) are two systems that aim to transfer routine works. To adapt to the trend of customizable and self-configuring production, factory workers often need to master new workflows, whereas machines and robots often need to be re-purposed, re-programmed, or even replaced. Unlike crowdsourcing or friendsourcing, factory workers often have higher level of determinism due to full-time employment and similar skills. They can be delegated with more complex workflows, if trained efficiently. AdapTutAR is developed to facilitate the training process in an adaptive way. Similarly, since machines and robots have the highest level of determinism, they can be delegated with many built-in tasks. To unleash the power of a network of robots and IoT machines, Vipo is introduced to enable requesters to program tasks that isolated machines cannot accomplish.

Through this dissertation, I contribute towards the goal of efficient task transfer by (i) identifying the latent structure of both creative and routine tasks as well as the characteristics of both human and machine workers, and (ii) developing systems and/or workflows to support proper task division and efficient communication accordingly.

One of the key takeaways is to divide a task as self-contained subtasks. For example, in BlueSky, generating ideas for a topic, generating dimensions/values, and classifying ideas based on dimensions/values are all self-contained. Consequently, crowd workers can join and leave in high frequency. In CoStory, requesters can cluster relevant story panels as a self-contained unit of task to be delegated. In such case, workers (friendsourcing) can contribute just within the given context. Similarly, in AdapTutAR, each step of machine operation task

is verifiable (being able to verify if the machine state is changed to expected state) and thus self-contained. Therefore, we can build deep learning models to determine if workers have done the step correctly. Lastly, subtasks of robots and machines is naturally the built-in capabilities of the devices. Hence, Vipo can be built to connect each subtask into a more flexible and complex workflow.

Another takeaway is to leverage scalablility to enhance efficiency. All four systems imply a one-to-many delegation pattern. In BlueSky, crowdsourcing–which is well-known for parallel tasks–is used to enlist ideas simultaneously. In CoStory, requesting alternative designs from friendsourcing also benefits from potentially parallel contribution. In AdapTutAR, the training process can occur in parallel as well as in distributed locations. In Vipo, the programmed tasks can be deployed to different devices if one device is not currently available. If we have to make a comparison, the tasks in AdapTutAR and Vipo are less scalable than those in BlueSky and CoStory, mainly because (i) the former involves physical machines (or robots, or IoT machines) that are limited and potentially shared by different tasks, (ii) the former consists of subtasks that often need to maintain a sequential order.

7.1 Future Work

Amid the rapid developing AR and Artificial Intelligent (AI) technologies, especially the emerging commercially AR devices and the tide of deep learning, the basic concept of AR and AI has become prevalent to a larger population. These technologies open several opportunities for future task transfer.

1. Creative task with AI support. Recent advancements in Natural Language Processing nourish a wide range of content generation systems, such as STORIUM [190], Botnik [191], and GPT-3 [192]. After their release, users created a large variety of creative work from poetry [193] to fictions [194]. Those auto-generated content may be beneficial to act as stimulus for humans' creativity.

2. Creative task with crowdsourcing/friendsourcing and AI collaboration. Some conversational assistants have been built to take advantage of crowdsourcing and AI, such

as Evorus [195]. Also, social media (e.g., Twitter[1]) has been testbeds for machine-generated content [196]. It seems fruitful to adopt these crowd+AI architecture into creative task, such as story generation, by encouraging human users to participate in the content generation. Different incentives and motivations could be explored.

3. Physical task with collaborative heterogeneous workers, including humans, robots, IoT devices and other AI-augmented machines. Apart from visual programming adopted in Vipo, the tasks can be shared among those agents through AR interfaces (e.g., GhostAR [197]), as well as AI-based interfaces (e.g., voice command, gestures, and even brain-AI interface [198]). Multi-human-multi-machine workers could be remote, collocated, or a hybrid.

4. Task division and allocation based on skill matching of tasks and workers. The skills required by a task and the skills mastered by a worker can be assessed or estimated by virtue of AI. Therefore, task partition and allocation based on the desire skills enables more efficient delegation. The end goal is that task transfer could be as fluid as data transfer. On the other hand, those systems should help workers build a "skill ladder" (i.e., a ladder for skill development) [199], otherwise workers may be fed with endless work without advancement.

In this last chapter, I mainly focus on discussing the retrospectives about the different phases of this dissertation, and suggesting the possible research directions based on the emerging technologies. I can envision the task transfer will become more transparent, fluid, and efficient in the near future.

[1]↑https://twitter.com

REFERENCES

[1] J. G. March and H. A. Simon, "Organizations," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 1496194, 1958. [Online]. Available: https://papers.ssrn.com/abstract=1496194.

[2] P. R. Lawrence and J. W. Lorsch, *Organization and Environment*. Harvard Business School Press, Jan. 15, 1967. [Online]. Available: https://www.hbs.edu/faculty/Pages/item.aspx?num=7917.

[3] T. W. Malone, T. W. Malone, and K. Crowston, "The interdisciplinary study of coordination," *ACM Comput. Surv.*, vol. 26, no. 1, pp. 87–119, Mar. 1994, ISSN: 0360-0300. DOI: 10.1145/174666.174668. [Online]. Available: http://doi.acm.org/10.1145/174666.174668.

[4] U. Melin, "Koordination och informationssystem i företag och nätverk," Ph.D. dissertation, Linköpings universitet, Linköping, Sweden, 2002. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-8515.

[5] M. Loskyll, I. Heck, J. Schlick, and M. Schwarz, "Context-based orchestration for control of resource-efficient manufacturing processes," *Future Internet*, vol. 4, no. 3, pp. 737–761, 2012, Publisher: Molecular Diversity Preservation International.

[6] D. Gorecky, M. Schmitt, M. Loskyll, and D. Zühlke, "Human-machine-interaction in the industry 4.0 era," in *2014 12th IEEE international conference on industrial informatics (INDIN)*, IEEE, 2014, pp. 289–294.

[7] G. Huang and A. J. Quinn, "BlueSky: Crowd-powered uniform sampling of idea spaces," in *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*, ser. C&C '17, event-place: Singapore, Singapore, New York, NY, USA: ACM, 2017, pp. 119–130, ISBN: 978-1-4503-4403-6. DOI: 10.1145/3059454.3059481. [Online]. Available: http://doi.acm.org/10.1145/3059454.3059481.

[8] G. Huang, M.-h. Wu, and A. J. Quinn, "Task design for crowdsourcing complex cognitive skills," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems Extended Abstracts*, ser. CHI '21 Extended Abstracts, event-place: Yokohama, Japan, New York, NY, USA: Association for Computing Machinery, May 8, 2021, pp. 1–7, ISBN: 978-1-4503-8096-6. DOI: 10.1145/3411763.3443447. [Online]. Available: http://doi.org/10.1145/3411763.3443447.

[9] G. Huang, P. S. Rao, M.-H. Wu, X. Qian, S. Y. Nof, K. Ramani, and A. J. Quinn, "Vipo: Spatial-visual programming with functions for robot-IoT workflows," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI '20, event-place: Honolulu, HI, USA, New York, NY, USA: Association for Computing Machinery, Apr. 21, 2020, pp. 1–13, ISBN: 978-1-4503-6708-0. DOI: 10.1145/3313831.3376670. [Online]. Available: http://doi.org/10.1145/3313831.3376670.

[10] G. Huang, X. Qian, T. Wang, F. Patel, M. Sreeram, Y. Cao, K. Ramani, and A. J. Quinn, "AdapTutAR: An adaptive tutoring system for machine tasks in augmented reality," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21, event-place: Yokohama, Japan, New York, NY, USA: Association for Computing Machinery, May 8, 2021, pp. 1–15, ISBN: 978-1-4503-8096-6. DOI: 10.1145/3411764.3445283. [Online]. Available: http://doi.org/10.1145/3411764.3445283.

[11] E. Barka and R. Sandhu, "Framework for role-based delegation models," in *Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00)*, Dec. 2000, pp. 168–176. DOI: 10.1109/ACSAC.2000.898870.

[12] Z. Liu, "A flexible role-based delegation model and its application in healthcare information system," Ph.D. dissertation, University of Toledo, 2013, 72 pp.

[13] E. Trist, "Collaboration in work settings: A personal perspective," *The Journal of Applied Behavioral Science*, vol. 13, no. 3, pp. 268–278, Jul. 1, 1977, ISSN: 0021-8863. DOI: 10.1177/002188637701300303. [Online]. Available: https://doi.org/10.1177/002188637701300303.

[14] M. B. Pinto, J. K. Pinto, and J. E. Prescott, "Antecedents and consequences of project team cross-functional cooperation," *Management Science*, vol. 39, no. 10, pp. 1281–1297, 1993, ISSN: 0025-1909. [Online]. Available: https://www.jstor.org/stable/2632967.

[15] A. Griffin and J. R. Hauser, "Integrating r&d and marketing: A review and analysis of the literature," *Journal of Product Innovation Management*, vol. 13, no. 3, pp. 191–215, May 1, 1996, ISSN: 0737-6782. DOI: 10.1016/0737-6782(96)00025-2. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0737678296000252.

[16] R. Larsson, *Coordination of Action in Mergers and Acquisitions: Interpretive and Systems Approaches Towards Synergy*. Lund University Press, 1990, 337 pp., Google-Books-ID: 74hJPQAACAAJ, ISBN: 978-0-86238-273-5.

[17] L. Taxén, "A framework for the coordination of complex systems' development," Ph.D. dissertation, Linköpings universitet, Linköping, Sweden, 2003. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-5001.

[18] J. Howe. (2006). "Crowdsourcing: A definition," Crowdsourcing, [Online]. Available: https://www.crowdsourcing.com.

[19] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut, "CrowdForge: Crowdsourcing complex work," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '11, New York, NY, USA: ACM, 2011, pp. 43–52, ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047202. [Online]. Available: http://doi.acm.org/10.1145/2047196.2047202.

[20] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich, "Soylent: A word processor with a crowd inside," in *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '10, New York, NY, USA: ACM, 2010, pp. 313–322, ISBN: 978-1-4503-0271-5. DOI: 10.1145/1866029.1866078. [Online]. Available: http://doi.acm.org/10.1145/1866029.1866078.

[21] M. A. Valentine, D. Retelny, A. To, N. Rahmati, T. Doshi, and M. S. Bernstein, "Flash organizations: Crowdsourcing complex work by structuring crowds as organizations," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17, event-place: Denver, Colorado, USA, New York, NY, USA: ACM, 2017, pp. 3523–3537, ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025811. [Online]. Available: http://doi.acm.org/10.1145/3025453.3025811.

[22] M. S. Bernstein, D. Tan, G. Smith, M. Czerwinski, and E. Horvitz, "Personalization via friendsourcing," *ACM Trans. Comput.-Hum. Interact.*, vol. 17, no. 2, 6:1–6:28, May 2008, ISSN: 1073-0516. DOI: 10.1145/1746259.1746260. [Online]. Available: http://doi.acm.org/10.1145/1746259.1746260.

[23] M. S. Bernstein, "Crowd-powered systems," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 2012, 237 pp.

[24] K. Moon, P. Bergemann, D. Brown, A. Chen, J. Chu, E. Eisen, G. Fischer, P. K. Loyalka, S. Rho, and J. Cohen, "Manufacturing productivity with worker turnover," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 3248075, Nov. 18, 2019. DOI: 10.2139/ssrn.3248075. [Online]. Available: https://papers.ssrn.com/abstract=3248075.

[25] I. Lee and K. Lee, "The internet of things (IoT): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, Jul. 1, 2015, ISSN: 0007-6813. DOI: 10.1016/j.bushor.2015.03.008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0007681315000373.

[26] P. P. Ray, "Internet of robotic things: Concept, technologies, and challenges," *IEEE Access*, vol. 4, pp. 9489–9500, 2016. DOI: 10.1109/ACCESS.2017.2647747.

[27] L. A. Grieco, A. Rizzo, S. Colucci, S. Sicari, G. Piro, D. Di Paola, and G. Boggia, "IoT-aided robotics applications: Technological implications, target domains and open issues," *Computer Communications*, vol. 54, pp. 32–47, Dec. 1, 2014, ISSN: 0140-3664. DOI: 10.1016/j.comcom.2014.07.013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366414002783.

[28] A. Grau, M. Indri, L. L. Bello, and T. Sauter, "Industrial robotics in factory automation: From the early stage to the internet of things," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, Oct. 2017, pp. 6159–6164. DOI: 10.1109/IECON.2017.8217070.

[29] T. Blecker and N. Abdelkafi, "Mass customization: State-of-the-art and challenges," in *Mass Customization: Challenges and Solutions*, ser. International Series in Operations Research & Management Science, T. Blecker and G. Friedrich, Eds., Boston, MA: Springer US, 2006, pp. 1–25, ISBN: 978-0-387-32224-7. DOI: . [Online]. Available: .

[30] L. Yu, A. Kittur, and R. E. Kraut, "Searching for analogical ideas with crowds," in *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '14, event-place: Toronto, Ontario, Canada, New York, NY, USA: ACM, 2014, pp. 1225–1234, ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557378. [Online]. Available: http://doi.acm.org/10.1145/2556288.2557378.

[31] L. Yu, A. Kittur, and R. E. Kraut, "Distributed analogical idea generation: Inventing with crowds," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14, New York, NY, USA: ACM, 2014, pp. 1245–1254, ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557371. [Online]. Available: http://doi.acm.org/10.1145/2556288.2557371.

[32] A. Parameswaran, M. H. Teh, H. Garcia-Molina, and J. Widom, "DataSift: An expressive and accurate crowd-powered search toolkit," in *First AAAI Conference on Human Computation and Crowdsourcing*, Nov. 3, 2013. [Online]. Available: https://www.aaai.org/ocs/index.php/HCOMP/HCOMP13/paper/view/7500.

[33] B. Trushkowsky, T. Kraska, M. J. Franklin, and P. Sarkar, "Crowdsourced enumeration queries," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, ISSN: 1063-6382, 1063-6382, Apr. 2013, pp. 673–684. DOI: 10.1109/ICDE.2013.6544865.

[34] J. O. Talton, D. Gibson, L. Yang, P. Hanrahan, and V. Koltun, "Exploratory modeling with collaborative design spaces," in *ACM SIGGRAPH Asia 2009 Papers*, ser. SIGGRAPH Asia '09, event-place: Yokohama, Japan, New York, NY, USA: ACM, 2009, 167:1–167:10, ISBN: 978-1-60558-858-2. DOI: 10.1145/1661412.1618513. [Online]. Available: http://doi.acm.org/10.1145/1661412.1618513.

[35] S. Chaudhuri, E. Kalogerakis, S. Giguere, and T. Funkhouser, "Attribit: Content creation with semantic attributes," in *Proceedings of the 26th annual ACM symposium on User interface software and technology*, Citation Key Alias: chaudhuri_attribit:_2013-1, ACM, 2013, pp. 193–202. [Online]. Available: http://dl.acm.org/citation.cfm?id=2502008.

[36] G. Smith, "Morphological charts: A systematic exploration of qualitative design space," Ph.D. dissertation, Clemson University, 2007. [Online]. Available:
.

[37] G. Smith, J. Richardson, J. D. Summers, and G. M. Mocko, "Concept exploration through morphological charts: An experimental study," *Journal of mechanical design*, vol. 134, no. 5, p. 051 004, 2012. [Online]. Available: http://heattransfer.asmedigitalcollection.asme.org/article.aspx?articleid=1450806.

[38] G. Pahl and W. Beitz, *Engineering design: a systematic approach*. Springer Science & Business Media, 2013. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=4uvSBwAAQBAJ&oi=fnd&pg=PR12&dq=Engineering+Design&ots=pkskqbtkSs&sig=vMc9VNNUtZZih1KRRmvq3tTOQ9w.

[39] M. J. French, *Conceptual Design for Engineers*, 3rd ed. London: Springer-Verlag, 1999, ISBN: 978-1-85233-027-9. DOI: 10.1007/978-1-4471-3627-9. [Online]. Available: https://www.springer.com/gp/book/9781852330279.

[40] N. Cross, *Engineering Design Methods: Strategies for Product Design (4th ed.)* Chichester: John Wiley & Sons, Apr. 2008, 230 pp., ISBN: 978-0-470-51926-4. [Online]. Available: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470519266.html.

[41] K. T. Ulrich, *Product Design and Development*. McGraw-Hill Education (India) Pvt Limited, Nov. 1, 2003, 388 pp., Google-Books-ID: CfKlfk_uhzoC, ISBN: 978-0-07-058513-3.

[42] G. Q. Huang and K.-L. Mak, "Web-based morphological charts for concept design in collaborative product development," *Journal of Intelligent Manufacturing*, vol. 10, no. 3, pp. 267–278, 1999. [Online]. Available: http://link.springer.com/article/10.1023/A:1008999908120.

[43] C. L. Dym and P. Little, *Engineering Design: A Project Based Introduction*, 3 edition. Hoboken, N.J. : Chichester: Wiley, Aug. 18, 2008, 352 pp., ISBN: 978-0-470-22596-7.

[44] B. Shneiderman, "Creativity support tools: Accelerating discovery and innovation," *Commun. ACM*, vol. 50, no. 12, pp. 20–32, Dec. 2007, ISSN: 0001-0782. DOI: 10.1145/1323688.1323689. [Online]. Available: http://doi.acm.org/10.1145/1323688.1323689.

[45] L. Yu and J. V. Nickerson, "Cooks or cobblers?: Crowd creativity through combination," in *Proceedings of the SIGCHI conference on human factors in computing systems*, ACM, 2011, pp. 1393–1402. [Online]. Available: http://dl.acm.org/citation.cfm?id=1979147.

[46] L. Yu and Y. Sakamoto, "Feature selection in crowd creativity," in *Foundations of Augmented Cognition. Directing the Future of Adaptive Systems*, ser. Lecture Notes in Computer Science 6780, D. D. Schmorrow and C. M. Fidopiastis, Eds., Springer Berlin Heidelberg, 2011, pp. 383–392, ISBN: 978-3-642-21852-1. [Online]. Available: .

[47] J. Preece, H. Sharp, and Y. Rogers, *Interaction Design: Beyond Human-Computer Interaction*. Wiley, 2015, ISBN: 978-1-119-02075-2. [Online]. Available: https://books.google.com/books?id=n0h9CAAAQBAJ.

[48] E. Arias, H. Eden, G. Fischer, A. Gorman, and E. Scharff, "Transcending the individual human mind—creating shared understanding through collaborative design," *ACM Trans. Comput.-Hum. Interact.*, vol. 7, no. 1, pp. 84–113, Mar. 2000, ISSN: 1073-0516. DOI: 10.1145/344949.345015. [Online]. Available: http://doi.acm.org/10.1145/344949.345015.

[49] D. C. Engelbart, "Toward augmenting the human intellect and boosting our collective IQ," *Commun. ACM*, vol. 38, no. 8, pp. 30–32, Aug. 1995, ISSN: 0001-0782. DOI: 10.1145/208344.208352. [Online]. Available: http://doi.acm.org/10.1145/208344.208352.

[50] B. Atasoy and J.-B. Martens, "STORYPLY: Designing for user experiences using storycraft," in *Collaboration in Creative Design*, Springer, Cham, 2016, pp. 181–210, ISBN: 978-3-319-29155-0. DOI: . [Online]. Available: .

[51] M. Haesen, D. Vanacken, K. Luyten, and K. Coninx, "Storyboards as a lingua franca in multidisciplinary design teams," in *Collaboration in Creative Design*, Springer, Cham, 2016, pp. 211–231, ISBN: 978-3-319-29155-0. DOI: . [Online]. Available: .

[52] J. Hoshino and Y. Hoshino, "Intelligent storyboard for prototyping animation," in *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*, Aug. 2001, pp. 377–380. DOI: 10.1109/ICME.2001.1237735.

[53] M. Haesen, J. Meskens, K. Luyten, and K. Coninx, "Draw me a storyboard: Incorporating principles & techniques of comics...," in *Proceedings of the 24th BCS Interaction Specialist Group Conference*, ser. BCS '10, Swinton, UK, UK: British Computer Society, 2010, pp. 133–142, ISBN: 978-1-78017-130-2. [Online]. Available: http://dl.acm.org/citation.cfm?id=2146303.2146323.

[54] J. T. Hackos and J. C. Redish, *User and Task Analysis for Interface Design*. New York, NY, USA: John Wiley & Sons, Inc., 1998, ISBN: 978-0-471-17831-6.

[55] C. Van der Lelie, "The value of storyboards in the product design process," *Personal and ubiquitous computing*, vol. 10, no. 2, pp. 159–162, 2006. [Online]. Available: http://link.springer.com/article/10.1007/s00779-005-0026-7.

[56] A. Tanaka, J. Craighead, G. Taylor, and R. Sottilare, "Adaptive learning technology for AR training: Possibilities and challenges," in *Adaptive Instructional Systems*, R. A. Sottilare and J. Schwarz, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2019, pp. 142–150, ISBN: 978-3-030-22341-0. DOI: .

[57] R. Sottilare and K. Brawner, "Component interaction within the generalized intelligent framework for tutoring (GIFT) as a model for adaptive instructional system standards toward standardization through design goals," pp. 55–62, 2018, ISBN: 0000000252782.

[58] J. Lee and O. Park, "Adaptive instructional systems," *Handbook of research on educational communications and technology*, pp. 469–484, 2008, Publisher: Citeseer.

[59] C. Froschl, "User modeling and user profiling in adaptive e-learning systems," *Graz, Austria: Master Thesis*, 2005, Publisher: Citeseer.

[60] K. VanLehn, "The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems," *Educational Psychologist*, vol. 46, no. 4, pp. 197–221, 2011, Publisher: Routledge, ISSN: 0046-1520. DOI: 10.1080/00461520.2011.611369.

[61] P. J. Durlach, "Fundamentals, flavors, and foibles of adaptive instructional systems," in *Adaptive Instructional Systems*, R. A. Sottilare and J. Schwarz, Eds., Cham: Springer International Publishing, 2019, pp. 76–95, ISBN: 978-3-030-22341-0.

[62] B. Goldberg, K. Brawner, R. Sottilare, R. Tarr, D. R. Billings, and N. Malone, "Use of evidence-based strategies to enhance the extensibility of adaptive tutoring technologies," p. 12, 2012.

[63] D. L. Lusk, A. D. Evans, T. R. Jeffrey, K. R. Palmer, C. S. Wikstrom, and P. E. Doolittle, "Multimedia learning and individual differences: Mediating the effects of working memory capacity with segmentation," *British Journal of Educational Technology*, vol. 40, no. 4, pp. 636–651, 2009. DOI: 10.1111/j.1467-8535.2008.00848.x.

[64] M. H. S. B. Smits, J. Boon, D. M. A. Sluijsmans, and T. v. Gog, "Content and timing of feedback in a web-based learning environment: Effects on learning as a function of prior knowledge," *Interactive Learning Environments*, vol. 16, no. 2, pp. 183–193, 2008, Publisher: Routledge _eprint: https://doi.org/10.1080/10494820701365952. DOI: 10.1080/10494820701365952. [Online]. Available: https://doi.org/10.1080/10494820701365952.

[65] A. T. Corbett, K. R. Koedinger, and J. R. Anderson, "Chapter 37 - intelligent tutoring systems," in *Handbook of Human-Computer Interaction (Second Edition)*, M. G. Helander, T. K. Landauer, and P. V. Prabhu, Eds., Second Edition, Amsterdam: North-Holland, 1997, pp. 849–874, ISBN: 978-0-444-81862-1. DOI: 10.1016/B978-044481862-1.50103-5. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780444818621501035.

[66] M. A. Orey and W. A. Nelson, "Development principles for intelligent tutoring systems: Integrating cognitive theory into the development of computer-based instruction," *Educational Technology Research and Development*, vol. 41, no. 1, pp. 59–72, 1993, ISSN: 1556-6501. DOI: 10.1007/BF02297092. [Online]. Available: https://doi.org/10.1007/BF02297092.

[67] F. Gutierrez and J. Atkinson, "Adaptive feedback selection for intelligent tutoring systems," *Expert Systems with Applications*, vol. 38, no. 5, pp. 6146–6152, 2011, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2010.11.058. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417410012960.

[68] A. T. Bimba, N. Idris, A. Al-Hunaiyyan, R. B. Mahmud, and N. L. B. M. Shuib, "Adaptive feedback in computer-based learning environments: A review," *Adaptive Behavior*, vol. 25, no. 5, pp. 217–234, 2017. DOI: 10.1177/1059712317727590. [Online]. Available: https://doi.org/10.1177/1059712317727590.

[69] P. Brusilovsky and H.-D. Su, "Adaptive visualization component of a distributed web-based adaptive educational system," in *Intelligent Tutoring Systems*, S. A. Cerri, G. Gouardères, and F. Paraguaçu, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 229–238, ISBN: 978-3-540-47987-1.

[70] M. Beyyoudh, M. K. Idrissi, and S. Bennani, "A new approach of designing an intelligent tutoring system based on adaptive workflows and pedagogical games," in *2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*, Apr. 2018, pp. 1–7. DOI: 10.1109/ITHET.2018.8424619.

[71] W. S. Lages and D. A. Bowman, "Walking with adaptive augmented reality workspaces: Design and usage patterns," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, ser. IUI '19, Marina del Ray, California: Association for Computing Machinery, Mar. 17, 2019, pp. 356–366, ISBN: 978-1-4503-6272-6. DOI: 10.1145/3301275.3302278. [Online]. Available: https://doi.org/10.1145/3301275.3302278.

[72] G. Westerfield, A. Mitrovic, and M. Billinghurst, "Intelligent augmented reality training for motherboard assembly," *International Journal of Artificial Intelligence in Education*, vol. 25, no. 1, pp. 157–172, 2015, ISSN: 1560-4306. DOI: 10.1007/s40593-014-0032-x. [Online]. Available: https://doi.org/10.1007/s40593-014-0032-x.

[73] A. Kotranza, D. S. Lind, C. M. Pugh, and B. Lok, "Real-time in-situ visual feedback of task performance in mixed environments for learning joint psychomotor-cognitive tasks," in *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, Oct. 2009, pp. 125–134. DOI: 10.1109/ISMAR.2009.5336485.

[74] R. Sottilare, C. Ragusa, M. Hoffman, and B. Goldberg, "Characterizing an adaptive tutoring learning effect chain for individual and team tutoring," in *Proceedings of the Interservice/Industry Training Simulation & Education Conference, Orlando, Florida*, 2013.

[75] A. Fender, P. Herholz, M. Alexa, and J. Müller, "OptiSpace: Automated placement of interactive 3d projection mapping content," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18, event-place: Montreal QC, Canada, New York, NY, USA: Association for Computing Machinery, 2018, ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3173843. [Online]. Available: https://doi.org/10.1145/3173574.3173843.

[76] A. Fender, D. Lindlbauer, P. Herholz, M. Alexa, and J. Müller, "HeatSpace: Automatic placement of displays by empirical analysis of user behavior," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '17, Québec City, QC, Canada: Association for Computing Machinery, Oct. 20, 2017, pp. 611–621, ISBN: 978-1-4503-4981-9. DOI: 10.1145/3126594.3126621. [Online]. Available: https://doi.org/10.1145/3126594.3126621.

[77] A. L. Thomaz and C. Breazeal, "Teachable robots: Understanding human teaching behavior to build more effective robot learners," *Artificial Intelligence*, vol. 172, no. 6, pp. 716–737, 2008, ISSN: 00043702. DOI: 10.1016/j.artint.2007.09.009.

[78] D. Rodenburg, P. Hungler, S. A. Etemad, D. Howes, A. Szulewski, and J. Mclellan, "Dynamically adaptive simulation based on expertise and cognitive load," in *2018 IEEE Games, Entertainment, Media Conference (GEM)*, 2018, pp. 1–6.

[79] Y. Cao, X. Qian, T. Wang, R. Lee, K. Huo, and K. Ramani, "An exploratory study of augmented reality presence for tutoring machine tasks," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI '20, Honolulu, HI, USA: Association for Computing Machinery, Apr. 21, 2020, pp. 1–13, ISBN: 978-1-4503-6708-0. DOI: 10.1145/3313831.3376688. [Online]. Available: https://doi.org/10.1145/3313831.3376688.

[80] B. Herbert, B. Ens, A. Weerasinghe, M. Billinghurst, and G. Wigley, "Design considerations for combining augmented reality with intelligent tutors," *Computers and Graphics (Pergamon)*, vol. 77, pp. 166–182, 2018, Publisher: Elsevier Ltd, ISSN: 00978493. DOI: 10.1016/j.cag.2018.09.017. [Online]. Available: https://doi.org/10.1016/j.cag.2018.09.017.

[81] Z. Zhu, V. Branzoi, M. Wolverton, G. Murray, N. Vitovitch, L. Yarnall, G. Acharya, S. Samarasekera, and R. Kumar, "AR-mentor: Augmented reality based mentoring system," in *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2014, pp. 17–22.

[82] A. Reyes, O. Vergara, E. Bojórquez, V. Sánchez, and M. Nandayapa, "A mobile augmented reality system to support machinery operations in scholar environments: A MAR SYSTEM TO SUPPORT MACHINERY OPERATIONS IN SCHOLAR ENVIRONMENTS," *Computer Applications in Engineering Education*, 2016. DOI: 10.1002/cae.21772.

[83] J.-R. CHARDONNET, G. Fromentin, and J. OUTEIRO, "Augmented reality as an aid for the use of machine tools," in *15th Management and Innovative Technologies (MIT) Conference*, Sinaia, Romania, Sep. 2017, pp. 1–4. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01598613.

[84] S. W. Lawson and J. R. G. Pretlove, "Augmented reality for underground pipe inspection and maintenance," in *Telemanipulator and Telepresence Technologies V*, M. R. Stein, Ed., Backup Publisher: International Society for Optics and Photonics, vol. 3524, SPIE, 1998, pp. 98–104. DOI: 10.1117/12.333673. [Online]. Available: https://doi.org/10.1117/12.333673.

[85] J. Zhu, S. K. Ong, and A. Y. C. Nee, "A context-aware augmented reality assisted maintenance system," *International Journal of Computer Integrated Manufacturing*, vol. 28, no. 2, pp. 213–225, 2015, Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/0951192X.2013.874589. DOI: 10.1080/0951192X.2013.874589. [Online]. Available: https://doi.org/10.1080/0951192X.2013.874589.

[86] S. J. Henderson and S. Feiner, "Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret," in *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, 2009, pp. 135–144.

[87] F. D. Crescenzio, M. Fantini, F. Persiani, L. D. Stefano, P. Azzari, and S. Salti, "Augmented reality for aircraft maintenance training and operations support," *IEEE Computer Graphics and Applications*, vol. 31, no. 1, pp. 96–101, 2011.

[88] M. Yamashita and S. Sakane, "Adaptive annotation using a human-robot interface system PARTNER," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 3, 2001, 2661–2667 vol.3.

[89] S. Yonemoto, "Seamless annotation display for augmented reality," in *2013 International Conference on Cyberworlds*, 2013, pp. 387–387.

[90] A. F. Abate, V. Loia, M. Nappi, S. Ricciardi, and E. Boccola, "ASSYST: Avatar baSed SYStem mainTenance," *2008 IEEE Radar Conference, RADAR 2008*, no. 1, 2008, ISBN: 9781424415397. DOI: 10.1109/RADAR.2008.4720962.

[91] M. Funk, "Augmented reality at the workplace : A context-aware assistive system using in-situ projection," 2016. [Online]. Available: http://dx.doi.org/10.18419/opus-8997.

[92] S. Kim, G. Lee, W. Huang, H. Kim, W. Woo, and M. Billinghurst, "Evaluating the combination of visual communication cues for HMD-based mixed reality remote collaboration," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19, event-place: Glasgow, Scotland Uk, New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–13, ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300403. [Online]. Available: https://doi.org/10.1145/3290605.3300403.

[93] S. J. Henderson and S. K. Feiner, "Augmented reality in the psychomotor phase of a procedural task," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 191–200. DOI: 10.1109/ISMAR.2011.6092386.

[94] P. T. Chua, R. Crivella, B. Daly, Ning Hu, R. Schaaf, D. Ventura, T. Camill, J. Hodgins, and R. Pausch, "Training for physical tasks in virtual environments: Tai chi," in *IEEE Virtual Reality, 2003. Proceedings.*, ISSN: 1087-8270, Mar. 2003, pp. 87–94. DOI: 10.1109/VR.2003.1191125.

[95] P.-H. Han, Y.-S. Chen, Y. Zhong, H.-L. Wang, and Y.-P. Hung, "My tai-chi coaches: An augmented-learning tool for practicing tai-chi chuan," in *Proceedings of the 8th Augmented Human International Conference*, ser. AH '17, event-place: Silicon Valley, California, USA, New York, NY, USA: Association for Computing Machinery, 2017, ISBN: 978-1-4503-4835-5. DOI: 10.1145/3041164.3041194. [Online]. Available: https://doi.org/10.1145/3041164.3041194.

[96] T. Piumsomboon, G. A. Lee, J. D. Hart, B. Ens, R. W. Lindeman, B. H. Thomas, and M. Billinghurst, "Mini-me: An adaptive avatar for mixed reality remote collaboration," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18, Montreal QC, Canada: Association for Computing Machinery, Apr. 19, 2018, pp. 1–13, ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3173620. [Online]. Available: http://doi.org/10.1145/3173574.3173620.

[97] T. Piumsomboon, G. A. Lee, A. Irlitti, B. Ens, B. H. Thomas, and M. Billinghurst, "On the shoulder of the giant: A multi-scale mixed reality collaboration with 360 video sharing and tangible interaction," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19, event-place: Glasgow, Scotland Uk, New York, NY, USA: Association for Computing Machinery, 2019, ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300458. [Online]. Available: https://doi.org/10.1145/3290605.3300458.

[98] B. Thoravi Kumaravel, F. Anderson, G. Fitzmaurice, B. Hartmann, and T. Grossman, "Loki: Facilitating remote instruction of physical tasks using bi-directional mixed-reality telepresence," in *Proceedings of the 32Nd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '19, event-place: New Orleans, LA, USA, New York, NY, USA: ACM, 2019, pp. 161–174, ISBN: 978-1-4503-6816-2. DOI: 10.1145/3332165.3347872. [Online]. Available: http://doi.acm.org/10.1145/3332165.3347872.

[99] D. Lindlbauer, A. M. Feit, and O. Hilliges, "Context-aware online adaptation of mixed reality interfaces," in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '19, New Orleans, LA, USA: Association for Computing Machinery, Oct. 17, 2019, pp. 147–160, ISBN: 978-1-4503-6816-2. DOI: 10.1145/3332165.3347945. [Online]. Available: https://doi.org/10.1145/3332165.3347945.

[100] A. Wegerich and M. Rötting, "A context-aware adaptation system for spatial augmented reality," in *Digital Human Modeling*, V. G. Duffy, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 417–425, ISBN: 978-3-642-21799-9.

[101] M. Tatzgern, V. Orso, D. Kalkofen, G. Jacucci, L. Gamberini, and D. Schmalstieg, "Adaptive information density for augmented reality displays," in *Proceedings - IEEE Virtual Reality*, vol. 2016-July, Greenville, SC, United states, 2016, pp. 83–92. [Online]. Available: http://dx.doi.org/10.1109/VR.2016.7504691.

[102] M. Blackstock and R. Lea, "Toward a distributed data flow platform for the web of things (distributed node-RED)," in *Proceedings of the 5th International Workshop on Web of Things*, ser. WoT '14, event-place: Cambridge, MA, USA, New York, NY, USA: ACM, 2014, pp. 34–39, ISBN: 978-1-4503-3066-4. DOI: 10.1145/2684432.2684439. [Online]. Available: http://doi.acm.org/10.1145/2684432.2684439.

[103] B. Ens, F. Anderson, T. Grossman, M. Annett, P. Irani, and G. Fitzmaurice, "Ivy: Exploring spatially situated visual programming for authoring and understanding intelligent environments," in *Proceedings of the 43rd Graphics Interface Conference*, ser. GI '17, event-place: Edmonton, Alberta, Canada, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2017, pp. 156–162, ISBN: 978-0-9947868-2-1. DOI: 10.20380/GI2017.20. [Online]. Available: https://doi.org/10.20380/GI2017.20.

[104] M. Blackstock and R. Lea, "WoTKit: A lightweight toolkit for the web of things," in *Proceedings of the Third International Workshop on the Web of Things*, ser. WOT '12, event-place: Newcastle, United Kingdom, New York, NY, USA: ACM, 2012, 3:1–3:6, ISBN: 978-1-4503-1603-3. DOI: 10.1145/2379756.2379759. [Online]. Available: http://doi.acm.org/10.1145/2379756.2379759.

[105] E. Pot, J. Monceaux, R. Gelin, and B. Maisonnier, "Choregraphe: A graphical tool for humanoid robot programming," in *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, Sep. 2009, pp. 46–51. DOI: 10.1109/ROMAN.2009.5326209.

[106] J. Huang, T. Lau, and M. Cakmak, "Design and evaluation of a rapid programming system for service robots," in *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, ser. HRI '16, Piscataway, NJ, USA: IEEE Press, 2016, pp. 295–302, ISBN: 978-1-4673-8370-7. [Online]. Available: http://dl.acm.org/citation.cfm?id=2906831.2906883.

[107] J. F. Kelly, *Lego Mindstorms NXT-G Programming Guide*. Apress, 2010.

[108] J. Jackson, "Microsoft robotics studio: A technical introduction," *IEEE Robotics Automation Magazine*, vol. 14, no. 4, pp. 82–87, Dec. 2007, ISSN: 1070-9932. DOI: 10.1109/M-RA.2007.905745.

[109] J. P. Diprose, B. A. MacDonald, and J. G. Hosking, "Ruru: A spatial and interactive visual programming language for novice robot programming," in *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Sep. 2011, pp. 25–32. DOI: 10.1109/VLHCC.2011.6070374.

[110] J. Huang and M. Cakmak, "Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts," in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, Mar. 2017, pp. 453–462.

[111] C. Datta, C. Jayawardena, I. H. Kuo, and B. A. MacDonald, "RoboStudio: A visual programming environment for rapid authoring and customization of complex services on a personal service robot," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 2352–2357. DOI: 10.1109/IROS.2012.6386105.

[112] F. Steinmetz, A. Wollschläger, and R. Weitschat, "RAZER—a HRI for visual task-level programming and intuitive skill parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1362–1369, Jul. 2018, ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2798300.

[113] J. López, D. Pérez, and E. Zalama, "A framework for building mobile single and multi-robot applications," *Robotics and Autonomous Systems*, vol. 59, no. 3, pp. 151–162, Mar. 1, 2011, ISSN: 0921-8890. DOI: 10.1016/j.robot.2011.01.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092188901100011X.

[114] I. Zubrycki, M. Kolesiński, and G. Granosik, "Graphical programming interface for enabling non-technical professionals to program robots and internet-of-things devices," in *Advances in Computational Intelligence*, vol. 10306, Cham: Springer International Publishing, 2017, pp. 620–631, ISBN: 978-3-319-59147-6. DOI: . [Online]. Available: .

[115] F. Robotics. (2019). "Cloud robotics and automation: FetchCore from fetch robotics," Fetch Robotics, [Online]. Available: https://fetchrobotics.com/products-technology/cloud-robotics-fetchcore/.

[116] M. R. Reisinger, J. Schrammel, and P. Fröhlich, "Visual languages for smart spaces: End-user programming between data-flow and form-filling," in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Oct. 2017, pp. 165–169. DOI: 10.1109/VLHCC.2017.8103464.

[117] IFTTT. (2019). "IFTTT," [Online]. Available: https://ifttt.com.

[118] Google. (2019). "Blockly," Google Developers, [Online]. Available: https://developers.google.com/blockly/.

[119] Y. Cao, Z. Xu, F. Li, W. Zhong, K. Huo, and K. Ramani, "V.ra: An in-situ visual authoring system for robot-IoT task planning with augmented reality," in *Proceedings of the 2019 on Designing Interactive Systems Conference*, ser. DIS '19, event-place: San Diego, CA, USA, New York, NY, USA: ACM, 2019, pp. 1059–1070, ISBN: 978-1-4503-5850-7. DOI: 10.1145/3322276.3322278. [Online]. Available: http://doi.acm.org/10.1145/3322276.3322278.

[120] S. Conversy, J. Garcia, G. Buisan, M. Cousy, M. Poirier, N. Saporito, D. Taurino, G. Frau, and J. Debattista, "Vizir: A domain-specific graphical language for authoring and operating airport automations," in *UIST 2018, 31st ACM Symposium on User Interface Software and Technology*, ser. UIST '18 The 31st Annual ACM Symposium on User Interface Software and Technology, Berlin, Germany: ACM SIGCHI, Oct. 2018, Pages 261–273/ ISBN: 978–1–4503–5948–1. DOI: 10.1145/3242587.3242623. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01886335.

[121] R. H. Kazi, F. Chevalier, T. Grossman, and G. Fitzmaurice, "Kitty: Sketching dynamic and interactive illustrations," in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '14, event-place: Honolulu, Hawaii, USA, New York, NY, USA: ACM, 2014, pp. 395–405, ISBN: 978-1-4503-3069-5. DOI: 10.1145/2642918.2647375. [Online]. Available: http://doi.acm.org/10.1145/2642918.2647375.

[122] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)," *RFC*, vol. 7252, pp. 1–112, 2014. DOI: 10.17487/rfc7252.

[123] U. Hunkeler, H. L. Truong, and A. J. Stanford-Clark, "MQTT-s — a publish/subscribe protocol for wireless sensor networks," *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, pp. 791–798, 2008. DOI: 10.1109/COMSWA.2008.4554519.

[124] P. Saint-Andre, *Extensible messaging and presence protocol (XMPP): Core*, 2011. [Online]. Available: https://tools.ietf.org/html/rfc6120.

[125] L. Richardson and S. Ruby, *Restful Web Services*, First. O'Reilly, 2007, ISBN: 978-0-596-52926-0.

[126] S. Vinoski, "Advanced message queuing protocol," *IEEE Internet Computing*, vol. 10, no. 6, pp. 87–89, Nov. 2006, ISSN: 1089-7801, 1941-0131. DOI: 10.1109/MIC.2006.116.

[127] E. Miller, "An introduction to the resource description framework," *Bulletin of the American Society for Information Science and Technology*, vol. 25, no. 1, pp. 15–19, 1998, Citation Key Alias: miller_introduction_1998-1, ISSN: 1550-8366. DOI: 10.1002/bult.105. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/bult.105.

[128] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor, "IoT-lite: A lightweight semantic model for the internet of things," in *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Jul. 2016, pp. 90–97. DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0035.

[129] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, Kobe, Japan, 2009, p. 5.

[130] J. Howe, *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*, 1st ed. New York, NY, USA: Crown Publishing Group, 2008, ISBN: 978-0-307-39620-4.

[131] A. M. Koblin, "The sheep market," in *Proceedings of the Seventh ACM Conference on Creativity and Cognition*, ser. C&C '09, event-place: Berkeley, California, USA, New York, NY, USA: ACM, 2009, pp. 451–452, ISBN: 978-1-60558-865-0. DOI: 10.1145/1640233.1640348. [Online]. Available: http://doi.acm.org/10.1145/1640233.1640348.

[132] D. C. Brabham, "Crowdsourcing as a model for problem solving: An introduction and cases," *Convergence*, vol. 14, no. 1, pp. 75–90, Feb. 1, 2008, ISSN: 1354-8565. DOI: 10.1177/1354856507084420. [Online]. Available: https://doi.org/10.1177/1354856507084420.

[133] H. B. Abrams, "Originality and creativity in copyright law," *Law and Contemporary Problems*, no. 2, pp. 3–44, 1992. [Online]. Available: https://heinonline.org/HOL/P?h=hein.journals/lcp55&i=341.

[134] S. M. Smith, T. B. Ward, and J. S. Schumacher, "Constraining effects of examples in a creative generation task," *Memory & Cognition*, vol. 21, no. 6, pp. 837–845, Nov. 1993, ISSN: 0090-502X, 1532-5946. DOI: 10.3758/BF03202751. [Online]. Available: http://link.springer.com/article/10.3758/BF03202751.

[135] L. B. Chilton, G. Little, D. Edge, D. S. Weld, and J. A. Landay, "Cascade: Crowdsourcing taxonomy creation," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2013, pp. 1999–2008.

[136] A. Kulkarni, M. Can, and B. Hartmann, "Collaboratively crowdsourcing workflows with turkomatic," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, ser. CSCW '12, New York, NY, USA: ACM, 2012, pp. 1003–1012, ISBN: 978-1-4503-1086-4. DOI: 10.1145/2145204.2145354. [Online]. Available: http://doi.acm.org/10.1145/2145204.2145354.

[137] C. Haeussler and H. Sauermann, "The division of labor in teams: A conceptual framework and application to collaborations in science," National Bureau of Economic Research, Cambridge, MA, w22241, May 2016. DOI: 10.3386/w22241. [Online]. Available: http://www.nber.org/papers/w22241.pdf.

[138] S. Jabbehdari and J. P. Walsh, "Authorship norms and project structures in science," *Science, Technology, & Human Values*, vol. 42, no. 5, pp. 872–900, Sep. 1, 2017, ISSN: 0162-2439. DOI: 10.1177/0162243917697192. [Online]. Available: https://doi.org/10.1177/0162243917697192.

[139] F. Yoshikane, T. Nozawa, and K. Tsuji, "Comparative analysis of co-authorship networks considering authors' roles in collaboration: Differences between the theoretical and application areas," *Scientometrics*, vol. 68, no. 3, pp. 643–655, Sep. 1, 2006, ISSN: 1588-2861. DOI: 10.1007/s11192-006-0113-1. [Online]. Available: https://doi.org/10.1007/s11192-006-0113-1.

[140] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in GitHub: Transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, ser. CSCW '12, event-place: Seattle, Washington, USA Citation Key Alias: dabbish_social_2012, New York, NY, USA: ACM, 2012, pp. 1277–1286, ISBN: 978-1-4503-1086-4. DOI: 10.1145/2145204.2145396. [Online]. Available: http://doi.acm.org/10.1145/2145204.2145396.

[141] V. Bellotti and S. Bly, "Walking away from the desktop computer: Distributed collaboration and mobility in a product design team," in *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '96, New York, NY, USA: ACM, 1996, pp. 209–218, ISBN: 978-0-89791-765-0. DOI: 10.1145/240080.240256. [Online]. Available: http://doi.acm.org/10.1145/240080.240256.

[142] S. Greenberg, R. Blum, J. Dyck, K. Tee, and G. McEwan, "Supporting informal collaboration in shared-workspace groupware.," *J. UCS*, vol. 14, no. 9, pp. 1411–1434, 2008. [Online]. Available:

.

[143] F. Geyer, J. Budzinski, and H. Reiterer, "IdeaVis: A hybrid workspace and interactive visualization for paper-based collaborative sketching sessions," in *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, ser. NordiCHI '12, New York, NY, USA: ACM, 2012, pp. 331–340, ISBN: 978-1-4503-1482-4. DOI: 10.1145/2399016.2399069. [Online]. Available: http://doi.acm.org/10.1145/2399016.2399069.

[144] R. Gumienny, L. Gericke, M. Wenzel, and C. Meinel, "Supporting creative collaboration in globally distributed companies," in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, ser. CSCW '13, New York, NY, USA: ACM, 2013, pp. 995–1007, ISBN: 978-1-4503-1331-5. DOI: 10.1145/2441776.2441890. [Online]. Available: http://doi.acm.org/10.1145/2441776.2441890.

[145] S. D. Jap, ""pie sharing" in complex collaboration contexts," *Journal of Marketing Research*, vol. 38, no. 1, pp. 86–99, Feb. 1, 2001, ISSN: 0022-2437. DOI: 10.1509/jmkr.38.1.86.18827. [Online]. Available: https://doi.org/10.1509/jmkr.38.1.86.18827.

[146] J. Kim, J. Cheng, and M. S. Bernstein, "Ensemble: Exploring complementary strengths of leaders and crowds in creative collaboration," in *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, ser. CSCW '14, event-place: Baltimore, Maryland, USA, New York, NY, USA: ACM, 2014, pp. 745–755, ISBN: 978-1-4503-2540-0. DOI: 10.1145/2531602.2531638. [Online]. Available: http://doi.acm.org/10.1145/2531602.2531638.

[147] J. Schummer, "Multidisciplinarity, interdisciplinarity, and patterns of research collaboration in nanoscience and nanotechnology," *Scientometrics*, vol. 59, no. 3, pp. 425–465, Mar. 1, 2004, ISSN: 1588-2861. DOI: 10.1023/B:SCIE.0000018542.71314.38. [Online]. Available: https://doi.org/10.1023/B:SCIE.0000018542.71314.38.

[148] L. Lin, X. Geng, and A. B. Whinston, "A sender-receiver framework for knowledge transfer," *MIS Quarterly*, vol. 29, no. 2, pp. 197–219, 2005, ISSN: 0276-7783. DOI: 10.2307/25148677. [Online]. Available: https://www.jstor.org/stable/25148677.

[149] R. E. Kraut, R. S. Fish, R. W. Root, and B. L. Chalfonte, "Informal communication in organizations: Form, function, and technology," in *Human reactions to technology: Claremont symposium on applied social psychology*, 1990, pp. 145–199. [Online]. Available: https://pdfs.semanticscholar.org/60c5/25c2beee3d1820c020b7e880b46ca91f5685.pdf.

[150] S. Kristoffersen and F. Ljungberg, "An empirical study of how people establish interaction: Implications for CSCW session management models," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ACM, 1999, pp. 1–8. [Online]. Available: http://dl.acm.org/citation.cfm?id=302980.

[151] Z. Zhao, S. K. Badam, S. Chandrasegaran, D. G. Park, N. L. Elmqvist, L. Kisselburgh, and K. Ramani, "skWiki: A multimedia sketching system for collaborative creativity," in *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '14, New York, NY, USA: ACM, 2014, pp. 1235–1244, ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557394. [Online]. Available: http://doi.acm.org/10.1145/2556288.2557394.

[152] N. Mangano, T. D. LaToza, M. Petre, and A. van der Hoek, "Supporting informal design with interactive whiteboards," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14, New York, NY, USA: ACM, 2014, pp. 331–340, ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557411. [Online]. Available: http://doi.acm.org/10.1145/2556288.2557411.

[153] A. Tharatipyakul, H. Lee, S. Zhao, and R. C. Davis, "Supporting the comparison of alternative stories," in *Proceedings of the 28th Australian Conference on Computer-Human Interaction*, ser. OzCHI '16, New York, NY, USA: ACM, 2016, pp. 266–270, ISBN: 978-1-4503-4618-4. DOI: 10.1145/3010915.3010963. [Online]. Available: http://doi.acm.org/10.1145/3010915.3010963.

[154] R. Weber and S. Condoor, "Conceptual design using a synergistically compatible morphological matrix," in *FIE '98. 28th Annual Frontiers in Education Conference. Moving from 'Teacher-Centered' to 'Learner-Centered' Education. Conference Proceedings (Cat. No.98CH36214)*, ISSN: 0190-5848, vol. 1, Nov. 1998, 171–176 vol.1. DOI: 10.1109/FIE.1998.736828.

[155] J. W. Booth, A. K. Bhasin, T. Reid, and K. Ramani, "Evaluating the bottom-up method for functional decomposition in product dissection tasks," in *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers Digital Collection, Jan. 13, 2015. DOI: 10.1115/DETC2014-35393. [Online]. Available: https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings/IDETC-CIE2014/46346/V003T04A009/257288.

[156] H. M. Kim, N. F. Michelena, P. Y. Papalambros, and T. Jiang, "Target cascading in optimal system design," *Journal of Mechanical Design*, vol. 125, no. 3, pp. 474–480, Sep. 1, 2003, ISSN: 1050-0472. DOI: 10.1115/1.1582501. [Online]. Available: https://asmedigitalcollection.asme.org/mechanicaldesign/article/125/3/474/476050/Target-Cascading-in-Optimal-System-Design.

[157] C. Huxham, "Facilitating collaboration: Issues in multi-organizational group decision support in voluntary, informal collaborative settings," *Journal of the Operational Research Society*, vol. 42, no. 12, pp. 1037–1045, Dec. 1, 1991, ISSN: 0160-5682, 1476-9360. DOI: 10.1057/jors.1991.198. [Online]. Available: https://link.springer.com/article/10.1057/jors.1991.198.

[158] J. Hailpern, E. Hinterbichler, C. Leppert, D. Cook, and B. P. Bailey, "TEAM STORM: Demonstrating an interaction model for working with multiple ideas during creative group work," in *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition*, ser. C&C '07, New York, NY, USA: ACM, 2007, pp. 193–202, ISBN: 978-1-59593-712-4. DOI: 10.1145/1254960.1254987. [Online]. Available: http://doi.acm.org/10.1145/1254960.1254987.

[159] N. Kantola and T. Jokela, "SVSb: Simple and visual storyboards: Developing a visualisation method for depicting user scenarios," in *Proceedings of the 19th Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces*, ser. OZCHI '07, New York, NY, USA: ACM, 2007, pp. 49–56, ISBN: 978-1-59593-872-5. DOI: 10.1145/1324892.1324901. [Online]. Available: http://doi.acm.org/10.1145/1324892.1324901.

[160] M. Haller, M. Billinghurst, J. Leithinger, D. Leitner, and T. Seifried, "Coeno: Enhancing face-to-face collaboration," in *Proceedings of the 2005 International Conference on Augmented Tele-existence*, ser. ICAT '05, New York, NY, USA: ACM, 2005, pp. 40–47, ISBN: 978-0-473-10657-7. DOI: 10.1145/1152399.1152408. [Online]. Available: http://doi.acm.org/10.1145/1152399.1152408.

[161] R. Verganti and G. P. Pisano, "Which kind of collaboration is right for you?" *Harvard Business Review*, vol. 86, Dec. 1, 2008. [Online]. Available: https://www.hbs.edu/faculty/Pages/item.aspx?num=34824.

[162] D. G. Jansson and S. M. Smith, "Design fixation," *Design Studies*, vol. 12, no. 1, pp. 3–11, Jan. 1, 1991, ISSN: 0142-694X. DOI: 10.1016/0142-694X(91)90003-F. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0142694X9190003F.

[163] S. M. Smith and S. E. Blankenship, "Incubation and the persistence of fixation in problem solving," *The American Journal of Psychology*, vol. 104, no. 1, pp. 61–87, 1991, ISSN: 0002-9556.

[164] L. D. Xu, E. L. Xu, and L. Li, "Industry 4.0: State of the art and future trends," *International Journal of Production Research*, vol. 56, no. 8, pp. 2941–2962, 2018, Publisher: Taylor & Francis.

[165] P.-Y. Chi, S. Ahn, A. Ren, M. Dontcheva, W. Li, and B. Hartmann, "MixT: Automatic generation of step-by-step mixed media tutorials," in *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM, 2012, pp. 93–102. [Online]. Available: http://dl.acm.org/citation.cfm?id=2380130.

[166] M. Goto, Y. Uematsu, H. Saito, S. Senda, and A. Iketani, "Task support system by displaying instructional video onto AR workspace," in *2010 IEEE International Symposium on Mixed and Augmented Reality*, IEEE, 2010, pp. 83–90.

[167] E. Schoop, M. Nguyen, D. Lim, V. Savage, S. Follmer, and B. Hartmann, "Drill sergeant: Supporting physical construction projects through an ecosystem of augmented tools," in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2016, pp. 1607–1614.

[168] B. Thoravi Kumaravel, C. Nguyen, S. DiVerdi, and B. Hartmann, "TutoriVR: A video-based tutorial system for design applications in virtual reality," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–12.

[169] F. Winther, L. Ravindran, K. P. Svendsen, and T. Feuchtner, "Design and evaluation of a VR training simulation for pump maintenance based on a use case at grundfos," in *IEEE VR 2020*, IEEE, 2020.

[170] S. Maloney, M. Storr, S. Paynter, P. Morgan, and D. Ilic, "Investigating the efficacy of practical skill teaching: A pilot-study comparing three educational methods," *Advances in Health Sciences Education*, vol. 18, no. 1, pp. 71–80, 2013, Publisher: Springer.

[171] K. Kear, F. Chetwynd, J. Williams, and H. Donelan, "Web conferencing for synchronous online tutorials: Perspectives of tutors using a new medium," *Computers & Education*, vol. 58, no. 3, pp. 953–963, 2012, Publisher: Elsevier.

[172] J.-M. Hoc, "Towards a cognitive approach to human-machine cooperation in dynamic situations," *International Journal of Human-Computer Studies*, vol. 54, no. 4, pp. 509–540, Apr. 1, 2001, ISSN: 1071-5819. DOI: 10.1006/ijhc.2000.0454. [Online]. Available: https://doi.org/10.1006/ijhc.2000.0454.

[173] L. Suchman, *Human machine reconfigurations plans and situated actions*, 2nd. Cambridge University Press, 2007, 328 pp., ISBN: 978-0-521-85891-5.

[174] L.-E. Janlert, "The ubiquitous button," *Interactions*, vol. 21, no. 3, pp. 26–33, May 2014, Place: New York, NY, USA Publisher: Association for Computing Machinery, ISSN: 1072-5520. DOI: 10.1145/2592234. [Online]. Available: https://doi-org.ezproxy.lib.purdue.edu/10.1145/2592234.

[175] Facebook. (2021). "Oculus," [Online]. Available: https://www.oculus.com/.

[176] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf.

[177] P. Dvorak, R. Josth, and E. Delponte, "Object state recognition for automatic AR-based maintenance guidance," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, ISSN: 2160-7516, Jul. 2017, pp. 1244–1250. DOI: 10.1109/CVPRW.2017.164.

[178] M. Laielli, J. Smith, G. Biamby, T. Darrell, and B. Hartmann, "LabelAR: A spatial guidance interface for fast computer vision image collection," in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '19, New Orleans, LA, USA: Association for Computing Machinery, Oct. 17, 2019, pp. 987–998, ISBN: 978-1-4503-6816-2. DOI: 10.1145/3332165.3347927. [Online]. Available: https://doi.org/10.1145/3332165.3347927.

[179] Unity. (2021). "Unity real-time development platform," [Online]. Available: https://unity.com/.

[180] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[181] P. Martiskainen, M. Järvinen, J.-P. Skön, J. Tiirikainen, M. Kolehmainen, and J. Mononen, "Cow behaviour pattern recognition using a three-dimensional accelerometer and support vector machines," *Applied animal behaviour science*, vol. 119, no. 1, pp. 32–38, 2009, Publisher: Elsevier.

[182] Z.-Y. He and L.-W. Jin, "Activity recognition from acceleration data using AR model representation and SVM," in *2008 international conference on machine learning and cybernetics*, vol. 4, IEEE, 2008, pp. 2245–2250.

[183] A. Burova, J. Mäkelä, J. Hakulinen, T. Keskinen, H. Heinonen, S. Siltanen, and M. Turunen, "Utilizing VR and gaze tracking to develop AR solutions for industrial maintenance," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI '20, Honolulu, HI, USA: Association for Computing Machinery, Apr. 21, 2020, pp. 1–13, ISBN: 978-1-4503-6708-0. DOI: 10.1145/3313831.3376405. [Online]. Available: http://doi.org/10.1145/3313831.3376405.

[184] C. Archibald and E. Petriu, "Model for skills-oriented robot programming (SKORP)," in *Applications of Artificial Intelligence 1993: Machine Vision and Robotics*, vol. 1964, International Society for Optics and Photonics, Mar. 11, 1993, pp. 392–403. DOI: 10.1117/12.141787. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/1964/0000/Model-for-skills-oriented-robot-programming-SKORP/10.1117/12.141787.short.

[185] S. Bøgh, "Does your robot have skills?" *Proceedings of the 43rd International Symposium on Robotics*, 2012.

[186] M. R. Pedersen, D. L. Herzog, and V. Krüger, "Intuitive skill-level programming of industrial handling tasks on a mobile manipulator," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 4523–4530. DOI: 10.1109/IROS.2014.6943203.

[187] K. Huo, Y. Cao, S. H. Yoon, Z. Xu, G. Chen, and K. Ramani, "Scenariot: Spatially mapping smart things within augmented reality scenes," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18, New York, NY, USA: ACM, 2018, 219:1–219:13, ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3173793. [Online]. Available: http://doi.acm.org/10.1145/3173574.3173793.

[188] A. F. Blackwell and T. R. G. Green, "A cognitive dimensions questionnaire optimised for users," in *PPIG*, 2000.

[189] G. Morel, C. E. Pereira, and S. Y. Nof, "Historical survey and emerging challenges of manufacturing automation modeling and control: A systems architecting perspective," *Annual Reviews in Control*, vol. 47, pp. 21–34, Jan. 1, 2019, ISSN: 1367-5788. DOI: 10.1016/j.arcontrol.2019.01.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1367578818301263.

[190] N. Akoury, S. Wang, J. Whiting, S. Hood, N. Peng, and M. Iyyer, "STORIUM: A dataset and evaluation platform for machine-in-the-loop story generation," *arXiv:2010.01717 [cs]*, Oct. 4, 2020. arXiv: 2010.01717. [Online]. Available: http://arxiv.org/abs/2010.01717.

[191] Botnik. (). "Botnik – human-machine entertainment," [Online]. Available: https://botnik.org/.

[192] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *arXiv:2005.14165 [cs]*, Jul. 22, 2020. arXiv: 2005.14165. [Online]. Available: http://arxiv.org/abs/2005.14165.

[193] N. Köbis and L. D. Mossink, "Artificial intelligence versus maya angelou: Experimental evidence that people cannot differentiate AI-generated from human-written poetry," *Computers in Human Behavior*, vol. 114, p. 106 553, Jan. 1, 2021, ISSN: 0747-5632. DOI: 10.1016/j.chb.2020.106553. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747563220303034.

[194] G. Branwen, "GPT-3 creative fiction," Jun. 19, 2020, Last Modified: 2020-09-28. [Online]. Available: https://www.gwern.net/GPT-3.

[195] T.-H. (Huang, J. C. Chang, and J. P. Bigham, "Evorus: A crowd-powered conversational assistant built to automate itself over time," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18, New York, NY, USA: Association for Computing Machinery, Apr. 21, 2018, pp. 1–13, ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3173869. [Online]. Available: http://doi.org/10.1145/3173574.3173869.

[196] T. Veale and M. Cook, "Twitterbots: Making machines that make meaning," Sep. 11, 2018. DOI: 10.7551/mitpress/10859.001.0001. [Online]. Available: https://direct.mit.edu/books/book/4113/TwitterbotsMaking-Machines-that-Make-Meaning.

[197] Y. Cao, T. Wang, X. Qian, P. S. Rao, M. Wadhawan, K. Huo, and K. Ramani, "GhostAR: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality," in *Proceedings of the 32Nd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '19, event-place: New Orleans, LA, USA, New York, NY, USA: ACM, 2019, pp. 521–534, ISBN: 978-1-4503-6816-2. DOI: 10.1145/3332165.3347902. [Online]. Available: http://doi.acm.org/10.1145/3332165.3347902.

[198] X. Zhang, Z. Ma, H. Zheng, T. Li, K. Chen, X. Wang, C. Liu, L. Xu, X. Wu, D. Lin, and H. Lin, "The combination of brain-computer interfaces and artificial intelligence: Applications and challenges," *Annals of Translational Medicine*, vol. 8, no. 11, Jun. 2020, ISSN: 2305-5839. DOI: 10.21037/atm.2019.11.109. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7327323/.

[199] J. P. Bigham, K. Williams, N. Banerjee, and J. Zimmerman, "Scopist: Building a skill ladder into crowd transcription," in *Proceedings of the 14th International Web for All Conference*, ser. W4A '17, New York, NY, USA: Association for Computing Machinery, Apr. 2, 2017, pp. 1–10, ISBN: 978-1-4503-4900-0. DOI: 10.1145/3058555.3058562. [Online]. Available: https://doi.org/10.1145/3058555.3058562.

VITA

Gaoping Huang was born in Chongqing, China on 1991. He had studied at Xi'an Jiaotong University in Xi'an, Shaanxi, China during 2009–2013 and received bachelor's degree with a major in Electronic Science and Technology. During his undergraduate study, he gained initial research experience in non-linear optics and lasers. In 2014, he started his Ph.D study at Purdue University with a major in Electrical and Computer Engineering. His research interest centers around human-computer interaction, where he built web-based systems and AR/VR systems to enable efficient task delegation. For example, he built a crowdsourcing system that coordinates hundreds of online workers to generate creative ideas, a storyboarding system for eliciting interaction design ideas, a visual programming editor for Robot/IoT programming, a head-mounted AR system for tutoring machine tasks adaptively, and a mobile-phone based AR system for prototyping early-stage AR applications.