WeBuild: Automatically Distributing Assembly Tasks Among Collocated Workers to Improve Coordination

C. Ailie Fraser[1,2], Tovi Grossman[1], George Fitzmaurice[1]

[1]Autodesk Research
{tovi.grossman, george.fitzmaurice}@autodesk.com

[2]Design Lab, UC San Diego
cafraser@cs.ucsd.edu

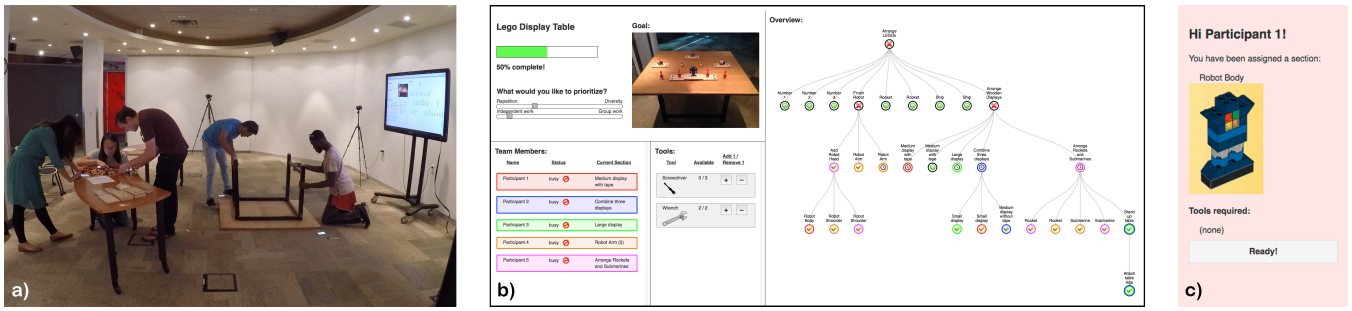


Figure 1: WeBuild helps collocated groups coordinate tasks. a) A group completes our study task with the help of WeBuild. b) A shared dashboard display provides a task overview. c) Each user views personalized step-by-step instructions on a mobile phone.

ABSTRACT

Physical construction and assembly tasks are often carried out by groups of collocated workers, and they can be difficult to coordinate. Group members must spend time deciding how to split up the task, how to assign subtasks to each other, and in what order subtasks should be completed. Informed by an observational study examining group coordination challenges, we built a task distribution system called WeBuild. Our custom algorithm dynamically assigns subtasks to workers in a group, taking into account factors such as the dependencies between subtasks and the skills of each group member. Each worker views personalized step-by-step instructions on a mobile phone, while a dashboard visualizes the entire process. An initial study found that WeBuild reduced the start-up time needed to coordinate and begin a task, and provides direction for future research to build on toward improving group efficiency and coordination for complex tasks.

Author Keywords

Task distribution; collaboration; coordination; assembly instructions

ACM Classification Keywords

H.5.3. Information interfaces and presentation (e.g., HCI): Group and Organization Interfaces.

CHI 2017, May 06-11, 2017, Denver, CO, USA
© 2017 ACM. ISBN 978-1-4503-4655-9/17/05...$15.00
DOI: http://dx.doi.org/10.1145/3025453.3026036

INTRODUCTION

Physical building tasks, such as building construction, furniture assembly, and toy kit construction, are often carried out by groups of collocated workers. With the exception of professional settings that involve a dedicated manager, the coordination amongst workers can be a challenge. The workers must spend time dividing and assigning tasks, locating and sharing tools, and figuring out how to execute instructions. As a result, the time spent actually completing the task may only be a fraction of the total time taken [44].

Traditional assembly instructions rarely indicate how multiple workers should collaborate (Figure 2a, b) or how the task can be efficiently subdivided (Figure 2c). While aids for physical assembly have been studied, they are often developed for a single user scenario (e.g. [5, 17, 18, 34, 38, 43, 45]). Multiple users have been considered, but typically with an expert providing remote assistance (e.g. [21, 23, 25, 33, 36]). With the exception of recent Crowdsourced Fabrication research [26], little work has looked at how to provide dynamic assistance to a team of collocated workers.

Our work aims to bring the known benefits of task management systems [27, 29, 39] and interactive instructions [1, 5, 17, 24] to the scenario of collocated group construction and assembly. We introduce WeBuild, a system that automatically distributes subtasks among workers, taking into account the workers' skills, the dependencies between subtasks, and the availability of tools, with the goal of improving group efficiency and coordination (Figure 1). WeBuild continuously adapts to the progress of the group to ensure an efficient distribution of tasks. An initial study found that WeBuild reduced the start-up time to coordinate and begin a task by 88%, and though further formal validation is needed, our findings suggest that such a system can improve group efficiency and satisfaction.

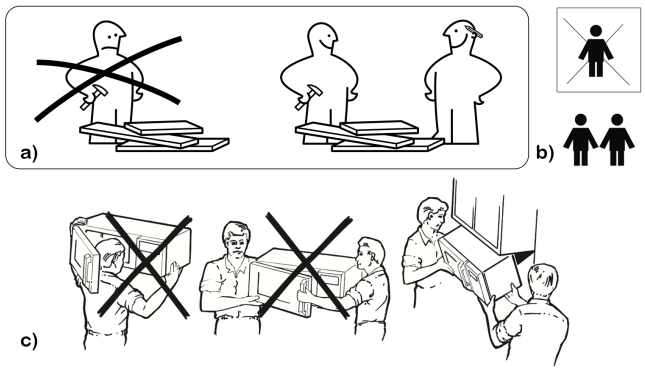


Figure 2: Instructions from an IKEA cabinet [46] **(a) and a JYSK shelf** [47] **(b) both show a 2-person requirement at the beginning, but do not show where or how the second person should help. c) Instructions for a microwave** [48] **show a second person helping for one step but no indication as to what this additional person should do for the rest of the task.**

The main contribution of our work is the design and implementation of WeBuild. We also contribute a back-end model for representing and subdividing physical task instructions, and an adaptive on-the-fly algorithm for distributing subtasks among workers. Finally we contribute observations and insights gathered from two studies, the first being an observational study of group coordination challenges, and the second being a preliminary evaluation of WeBuild as compared to a baseline condition of paper instructions. Our studies and implementations look at smaller-scale static tasks, such as toy building and furniture assembly, so we conclude by discussing how WeBuild could generalize to larger scales and more dynamically changing tasks, such as architectural construction and disaster relief.

RELATED WORK

Our work has been guided and inspired by previous research in the domains of physical task assistance, collaborative task aids, and automatic task distribution.

Real-time Assistance for Physical Tasks

Static instructions such as online tutorials and paper manuals are a popular resource for physical tasks. However, the burden is on the user to keep track of their progress and evaluate whether they are following the instructions correctly. Dynamic, context-sensitive assistance has been shown to help with these challenges by detecting user progress and stepping through instructions accordingly, and providing real-time feedback to ensure that users complete each step correctly [5, 17, 34, 43].

Researchers have explored several different modalities for displaying such real-time assistance for physical tasks. Wearable hands-free displays such as smartwatches have been found to be promising for providing short, relevant instructions [1, 26, 45]. Augmented reality, provided through projections or head-mounted displays, can further help to guide users through complex tasks [17, 18, 38, 43]. Alternatively, Knibbe et al. used a tabletop display to provide contextually relevant instructions for DIY tasks [24]. Our work similarly leverages both mobile devices and large displays, but adapts these concepts to a multi-user scenario.

In terms of what type of task to use for evaluating such assistive systems, Funk et al. proposed standardized benchmark tasks in two high-level categories: "pick-and-place" (e.g. LEGO) and "industrial assembly" (e.g. using screws and nails) [13]. The task used in our study is heavily based on these benchmarks but is designed specifically for a group of workers. Like Funk et al. [13, 14], we include our full task instructions as supplementary material to similarly serve as a potential benchmarking task or inspiration for future group assembly work.

Collaborative Task Aids

Most research examining collaboration on physical tasks has focused on improving *remote* collaboration, often between a worker and a remote expert [21, 23, 25, 33, 36]. For example, a remote helper's presence can be improved by displaying their hand gestures in 3D [36] or 2D video [23, 33], and hands-free technology can help users communicate while actively working on the task [21]. Our work focuses instead on *collocated* collaboration within a group of users, none of whom may be an expert at the task.

There are few examples of aids for collocated collaborative physical tasks. TurkDeck projects instructions onto the environment for a group of people arranging props in a haptic virtual reality scene [10]. Lafreniere et al.'s work on Crowdsourced Fabrication [26] advanced this to more complex tasks, where a crowd of workers assembled a large pavilion structure with dynamic assistance via smartwatches. However, the system had each worker complete the exact same task independently. This was done approximately two hundred times, with little required coordination or collaboration. Our system considers tasks comprising many different subtasks, each of which may involve different tools and require different numbers of people.

Collocated group collaboration has also been studied for other types of tasks such as brainstorming [11], decision-making [6], problem solving [9, 20, 37, 41], and software development [7]. While the main need of remote teams tends to be improved communication [7], collocated teams have been shown to mainly have need for improved task structuring and scheduling [8, 11], documentation of progress [7], and awareness of others' progress and overall task status [9, 32, 41]. Our work, informed by these findings, mainly focuses on task coordination and structuring, by automating the distribution and management of subtasks.

Task Distribution

The general problem of task distribution is found in many different domains, and it involves balancing a set of defined constraints, dependencies, and goals, the specifics of which depend on the domain at hand.

For example, with resource scheduling in operating systems, fair allocation is an important goal, and so randomized algorithms have been shown to work well [40]. For

software development and release planning, there are often many different potentially conflicting criteria to optimize, such as the various stakeholders' opinions [16]. Greer & Ruhe [16] presented a flexible algorithm that re-calculates the optimal plan after each increment is complete, thus adapting smoothly to changing circumstances.

Theoretically, this problem is known as the "resource-constrained project scheduling problem", and has been widely researched (e.g. [4, 31]). For example, Artigues showed that this problem can be framed as a combinatorial optimization problem [4]. Our work differs slightly in that we do not know the execution times of each subtask beforehand, but our formalization of the task model is similar.

In the industrial domain, physical assembly tasks are often carried out by groups of robot workers. A large body of work has focused on algorithms to optimally allocate tasks (e.g. [15, 35, 39]). Human workers tend to be less predictable, so these algorithms do not translate directly, though our own task model and algorithm were inspired by those used in the robotic task planning literature.

In summary, despite the extensive research on aids for physical assembly tasks, little work has been done to understand and aid in the challenges associated with coordinating a group of collocated workers for completing such tasks.

OBSERVATIONAL STUDY

To improve our understanding of the current challenges groups face and the strategies they use when collaborating on physical tasks, we conducted an exploratory observational study. To avoid constraining our findings and design goals to one specific task, we used two different tasks: two groups built a Meccano Tower Bridge set [49], and two groups built an IKEA cabinet [50].

Participants

We recruited 16 participants from our organization (4 female, ages 18 - 54) and split them into groups of four. All participants had at least some experience working with hand tools (mean 3.75/5, *SD* 0.86, where 1 = never used, and 5 = use all the time). Participants had limited experience working with IKEA and Meccano specifically: IKEA participants' mean experience was 2.63/5 (*SD* 0.74), and Meccano participants' mean was 1.25/5 (*SD* 0.46). In each group, some participants knew each other well, while others had never met.

Procedure

Each group was given four copies of the instruction booklets for their task. All of the required tools were provided, with enough tools for four people to work simultaneously on any individual step if they chose to do so. The IKEA task required screwdrivers and a hammer, and the Meccano task required a small wrench and a screwdriver.

Groups were told to accomplish as much of the task as they could within one hour. An experimenter took notes and observed group dynamics. After the session, each participant completed a questionnaire reflecting on how the session went and how they felt it could have been improved, as well as providing demographic information and details about their prior building experience. Observations and participant responses were analyzed to highlight recurrent themes and behaviours.

Observations

Task Distribution

All groups spent several minutes of initial "start-up time" figuring out how the different components of the task fit together and deciding how to split up the task. In both Meccano groups, each person worked on one of the four main towers. Both IKEA groups initially divided into two pairs, one building the cabinet body, and one building the four drawers (Figure 3). Once these main IKEA components were finished, additional time was spent figuring out which of the remaining components to do next. The distribution of the remaining subtasks varied between the two groups, as the optimal order to complete them in was not obvious.

Task distribution and collaboration strategies also varied between pairs. For those working on the IKEA drawers, one pair opted to build each drawer together one at a time, while the other opted to build the first drawer together, and the next two in parallel. The former group was more successful, as in the latter group, one participant worked ahead on their drawer while the other got stuck on an earlier step. Some time was lost before the first participant noticed and helped the second.

In the IKEA task, a few steps required two people to work together (e.g. lifting and positioning the large cabinet walls). However, we noticed that having two people work on a single-person subtask was often advantageous. For example, participants building the cabinet body would simultaneously screw in screws on opposite sides of the cabinet, eliminating the need for one participant to move around to both sides after completing each step.

Figure 3: A group in our observational study split into pairs to build the IKEA cabinet.

In general, once a person had completed a step or subtask once, they were able to do it again faster and more easily. Sometimes the group would recognize this and delegate a subtask to someone who had done it already. Several participants mentioned afterward that an "assembly line" setup might have been more efficient, as each group member could become a "specialist" at certain subtasks. However, figuring out how to distribute the tasks in this way would have taken longer as it would require a deeper understanding of the required subtasks from the onset.

Communication

Overall, communication was strong within pairs but weaker across the group. Participants working on the same subtask communicated frequently, asking each other questions, helping each other complete tricky steps, and gathering tools and parts for each other. There was less communication across the entire group, and several participants mentioned afterward that they wished they had been more aware of the status of the other pair on their team. This could potentially have helped with the challenge of task distribution, as not knowing what stage the other pair was at made it more difficult to decide what subtask to do next.

Prior Experience

Some participants seemed to be naturally suited for certain types of tasks. In some cases, this was due to prior experience (e.g. knowing how IKEA locking screws work). However, in other cases users exhibited inherent skills. One participant with smaller hands was better at a particularly finicky Meccano step. Another participant with no prior IKEA experience was better at assembling drawers than other participants who had some experience, which could be due to exceptional problem-solving or spatial reasoning skills.

DESIGN GOALS

Our observational study validated our belief that coordinating physical tasks in a group can be challenging and thus an automatic task distribution system would be useful. Based on our observations, and guided by our review of the literature, we derived a set of design goals for an intelligent task distribution system.

D1. Continuously Adapt to the Situation and Workers

While task allocation systems for robot workers can generate optimal distribution plans before the task has begun [39], a system for human workers must account for the unpredictability and individual differences of humans. It should therefore continuously adapt to the group's progress rather than determine a global schedule. For example, if one worker is faster at a certain subtask, the system should assign future similar subtasks to that worker. The system should also account for users' prior experience [27].

D2. Provide a Task Overview and Promote Awareness

Several participants mentioned a desire for more knowledge of the overall context of their current subtask, as well as more awareness of other group members' progress. This need for task and group awareness has also been well documented in the literature [8, 9, 41].

D3. Display Step-by-Step Instructions to Each Worker

As the literature on software tutorials (e.g. [12]) and physical task assistance (e.g. [24, 26]) has shown, it is beneficial for participants to see instructions one step at a time, to help them stay on track and know what is next without having to hunt through an instruction booklet.

D4. Support Adaptive Teamwork

Participants in our study frequently grouped into pairs, as having someone to talk to and compare work with was often helpful. An intelligent task distribution system should therefore support dynamic formation of sub-teams. The system should be aware of which subtasks should be done individually, and which can be aided by multiple workers, and assign subtasks accordingly.

WEBUILD: SYSTEM DESCRIPTION

Guided by the above design goals, we built a task distribution system called WeBuild. The goal of the WeBuild system is to help groups complete physical assembly tasks more efficiently, and to alleviate some of the coordination challenges that may otherwise be present. WeBuild takes in the step-by-step instructions for a task as input, including details about step dependencies, required tools, and associated skills, and it automatically assigns subtasks to workers. The algorithm works adaptively, only assigning the next subtask once a previous one is complete. The task instructions and details must be manually entered; our work assumes such information would be available, allowing us to focus on the system design and behaviour.

Range of Tasks Supported

WeBuild is designed to support assembly tasks that can be represented with *static* step-by-step instructions. These include group furniture building (e.g. assembling an IKEA kitchen), large art installations (e.g. pavilions), controlled construction projects (e.g. building a prefabricated house), or high-quantity manual tasks (e.g. circuit board assembly). It does not currently support *dynamic* tasks where the end goal may change or there are external uncontrollable variables (e.g. disaster relief and volunteer habitat building), though we aim to provide the groundwork for future research to explore these possibilities.

WeBuild is usable by both groups of workers who are familiar with each other, and groups of strangers. We predict that WeBuild will be particularly useful in situations where groups do not know each other (e.g. [26]) and are likely to find collaboration more challenging.

System Overview

WeBuild comprises a central server, a dashboard display, and a mobile smartphone for each group member (Figure 4). The central server can be any laptop or desktop in the work area. The system can scale to any number of workers, and can adapt to workers entering or exiting mid-task. The dashboard displays the overall process and structure of the task, to help support group awareness. Each worker's mobile phone displays instructions for their current subtask.

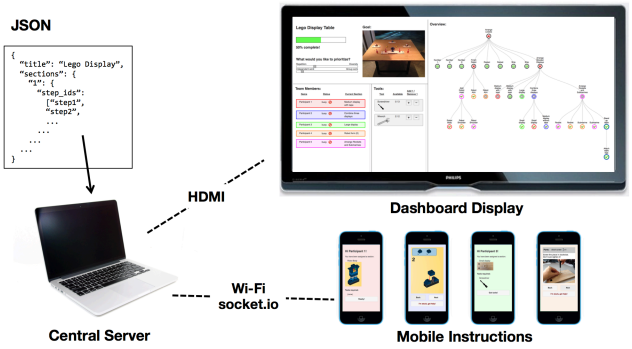


Figure 4: The WeBuild system architecture.

Task Initialization

Before the task can begin, a group member must specify how many of each required tool the group has available on the central server. Clicking "Start" begins the task, at which point the system begins allocating subtasks to all users.

Dashboard Display

The dashboard display provides high-level task information to promote group awareness (D2) (Figure 1b).

The left side displays a progress bar, parameter sliders, a photo of the final goal, and information about each user and tool (Figure 5). The "Team Members" section shows the names and statuses of all users, including which section of the task they are currently working on. The "Tools" section displays the tools required for the task, how many of each are in use or available, and buttons to update their quantity.

The right side of the dashboard displays the task overview in a tree representation, broken down into sections (Figure 6). The tree is structured by dependencies, with the root at the top being the final section, and the child nodes being the sections required for a parent node to be started.

Individual User View

Each group member uses a mobile phone to access an individual view (Figure 7). We chose mobile phones over smartwatches [1, 26] because of their ubiquity and larger screen real estate for displaying graphical instructions.

Each user connects to the IP address of the server, and logs in by entering their name, choosing a colour to be represented by, and rating their prior experience at each needed skill. Our algorithm uses this information to ensure that users are more likely to be assigned to tasks they are familiar with (D1).

The device informs the user when they are assigned to a section, and displays a photo of the section and its required tools (Figure 7a). Once the user has collected the tools and pressed the confirmation button, the instructions are shown one step at a time (Figure 7b). Users proceed through the steps manually using the "Next" and "Back" buttons (D3). Once the user completes the last step of the section, the system prompts them to confirm that they are finished, and displays a photo of the completed section to check their work against (Figure 7c).

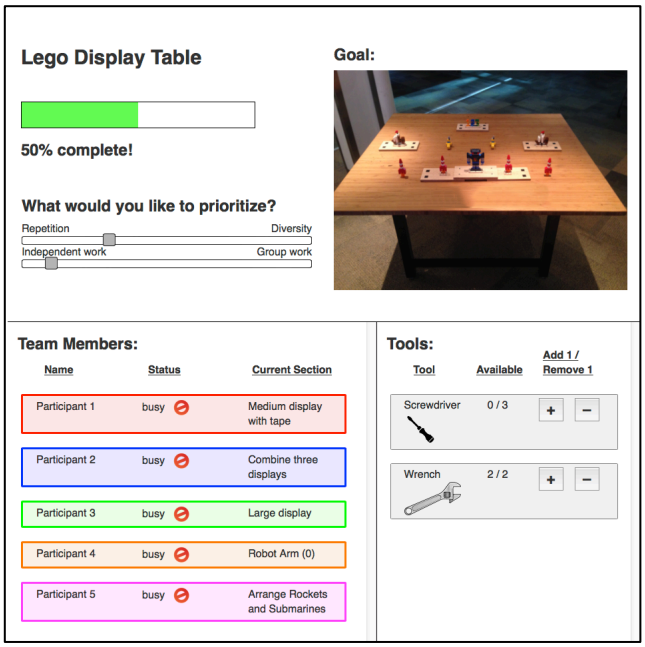


Figure 5: The left side of the dashboard displays a progress bar, parameter sliders, a photo of the final goal, and information about each user and the available tools.

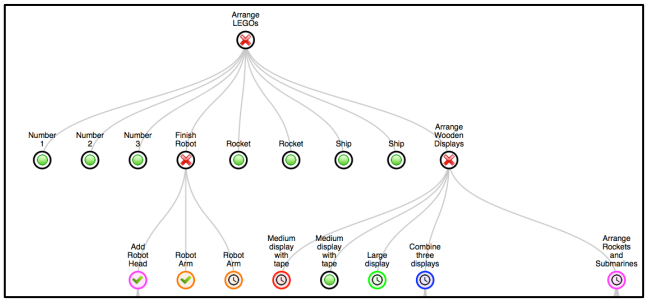


Figure 6: The right side of the dashboard displays a tree representation of the task. Green nodes have no dependencies and are therefore ready to be assigned. Red X nodes are waiting on their dependencies and therefore cannot be assigned yet. Completed nodes and nodes in progress show the colour(s) of the assigned worker(s).

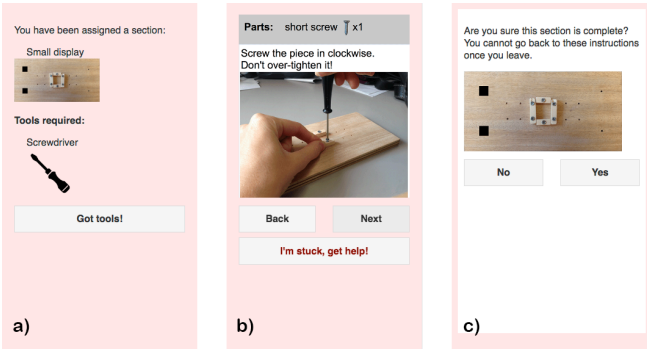


Figure 7: The WeBuild mobile interface. a) The user has been assigned a section and must gather the tools to start. b) The user is working on a section, and views each step one at a time. c) The user has finished their section, and must confirm it is complete.

When multiple users are assigned to work on a section together, the display shows who is on their sub-team and instructs them to find each other. Once one of the users presses the confirmation button to begin the section, all of the sub-team members' screens are synchronized to show the same step. For added validation, all members of the sub-team must confirm when the subtask has been completed.

If a user or group is stuck during a section, they can ask for help (Figure 7b). The next available user will then be assigned to join that section. At any time, a user can click "Exit" to leave the task entirely, at which point the system logs them out and reassigns the task they were working on.

Task Model

A prerequisite of our system is that a representation of the task already exists within the system. In our current implementation, this information is entered manually.

We developed a novel task representation based on similar models in the literature [4, 19, 27, 39]. A task T can be formally defined by the sets of sections, tools, and skills it is composed of:

$T = \{Sections, Tools, Skills\}$, where:

$Sections = \{S_1, S_2, ..., S_n\}$
$Tools = \{T_1, T_2, ..., T_m\}$
$Skills = \{K_1, K_2, ..., K_p\}$

Each Section S_i is a subtask, and is further defined by a sequence of steps from the task instructions ($Steps_i$), the tools required by each participant ($Tools_i$), the type of skills required ($Skills_i$), the minimum (Min_i) and maximum (Max_i) number of people that can work on the section, and the sections that must be complete for this section to begin ($Dependencies_i$):

$S_i = \{Steps_i, Tools_i, Skills_i, Max_i, Min_i, Dependencies_i\}$, where:

$Steps_i = \{step_{1i}, step_{2i}, ..., step_{qi}\}$
$Tools_i[j] = \{x \mid$ Tool T_x is required by the j^{th} user for section $S_i\}$
$Skills_i = \{y \mid$ Skill K_y is required for section $S_i\}$
$Dependencies_i = \{z \mid$ Section S_z is a dependency of section $S_i\}$

The skills required for the task could be defined at a high level, such as experience with woodworking, or at a lower level, such as experience with specific tools. Min_i is the minimum required number of people for a section, while Max_i is the maximum number of people that could be a useful addition (D4). The set of tools for each step ($Tools_i$) is a set of sets, as each subsequent worker may require a different set of tools. For example, if a second worker is required to hold a block while the first worker screws something in, then only the first worker needs a screwdriver.

For any two sections, we also calculate their similarity as a value between 0 and 1. In our current implementation, similarity is based on the number of common tools between the sections, to a maximum of 0.5, and is equal to 1 only if the sections are equivalent:

$$Similarity(i,j) = \begin{cases} 1, if\ S_i \stackrel{\text{def}}{=} S_j, \\ 0.5\dfrac{\|Tools_i \cap Tools_j\|}{Max(\|Tools_i\|, \|Tools_j\|)}, otherwise. \end{cases}$$

Task Distribution Algorithm

Our task distribution algorithm works on-the-fly: every time one or more users become available, it looks for the best section to assign to each available user (D1).

Algorithm Parameters

The algorithm includes two parameters whose values can vary between 0 and 1: these are *task diversity* and *group priority*. Both of these values can be adjusted with sliders on the dashboard at any time during the task.

If *task diversity* is 1, the system prioritizes giving users a variety of subtasks, which can increase engagement, produce higher quality work, and prevent repetitive strain [19, 29]. If *task diversity* is 0, workers are more likely to repeat the same or similar subtasks, which can help them learn specific skills and become more efficient [30].

If *group priority* is 1, the system biases towards assigning multiple workers to a section. If it is set to 0, the system prioritizes working individually. In cases where team building is particularly important, prioritizing group work may be preferred. In cases where it is important for workers to stay focused and quiet, individual work may be preferred.

Each time the algorithm is triggered, it prioritizes diversity with a probability equal to the value of the *task diversity* parameter. Otherwise, it prioritizes repetition. Similarly, it prioritizes group work with a probability equal to *group priority*. Otherwise, it prioritizes individual work.

Section Assignment Algorithm

The section assignment algorithm is triggered any time there is a free user. This happens when the task is started, when a new user joins, and when any section is completed. A detailed example of the values calculated by this algorithm is included in the Appendix. The algorithm generates a list of available users $U = \{u_1, u_2, ..., u_n\}$ who are not currently assigned to a section, and a list of available sections $A = \{S_1, S_2, ..., S_m\}$. A section is considered available if its required tools are available, it is currently unassigned or in progress with less than its maximum number of workers, and all its dependencies are complete. For each free user u_i, we build several lists P_i of priority values $\{p_{i1}, p_{i2}, ..., p_{iq}\}$ where p_{ij} is the priority value for assigning user u_i to section S_j. We build one list for each of the following criteria:

- **Tree depth (P_D):** $P_D[j]$ is the depth of section S_j in the task tree. Sections with larger depth are prioritized because the further a section is down the task tree, the more sections that are potentially waiting on it to be completed.
- **Group (P_G):** If individual work is being prioritized, $P_G[j]$ is 1 for all unassigned sections (as these do not yet have anyone working on them) and 0 for all sections in progress (where the user would be joining an existing team). Otherwise, the opposite values are set.

- **Prior Experience (P_E):** $P_E[j]$ is the user's self-rated experience (a value between 1 and 5) at the skill required by section S_j. If S_j requires multiple skills, it is the average of the user's experience ratings at those skills.

If the user has just completed a section S_c, we also create lists for the following criteria:

- **Next section (P_N):** $P_N[j]$ is 1 if section S_j is the next section after S_c (it is its parent node), and 0 otherwise. This prioritizes overall continuity of assignments.
- **Repetition or diversity (P_R):** If repetition is being prioritized, $P_R[j]$ is the similarity value *Similarity(j,c)*. Otherwise, $P_R[j]$ is set to 1 - *Similarity(j,c)*.
- **Speed (P_S):** This list prioritizes sections the user has already excelled at. For each available section S_j, we calculate the average time all users have taken on all completed sections weighted their by similarity to S_j, and the average time the current user has taken. $P_S[j]$ is the difference between these two averages.

To calculate the overall priority values P_{itotal} for each user u_i, we normalize each list above and add the values together element-wise.

$$P_{itotal} = P_{iD} + P_{iG} + P_{iE} + P_{iN} + P_{iR} + P_{iS}$$

We then use a greedy algorithm to make the section assignments: we find the single highest priority value in all of the P_{total} lists across every user, and assign that user to the corresponding section. We then recalculate the list of available sections and repeat the above process for the remaining free users, until all users have been assigned.

Initial Assignments

When the task is first started, we can only consider the first three criteria above, as no sections have been completed:

$$P_{itotal} = P_{iD} + P_{iG} + P_{iE}$$

However, we can still account for the *task diversity* slider value, by planning ahead. If repetition is being prioritized, we narrow down the list of available sections to a list of *unique* available sections, so that all users will be assigned to a different section, and thus more likely to repeat their section again later. Otherwise, we assign as many of the same section as possible to the available users so that each user will be less likely to repeat their section later.

Asking for Help

When a user presses the "get help" button, the section they are currently working on is added to a global list of sections that need help. The next time the algorithm runs, only sections in this help list are considered available. Each free user's priorities are calculated as above, and the user with the highest priority is assigned to help with that section.

Adapting to Users and Tools

Our algorithm provides the flexibility for users to join or leave the task at any point, by logging in or exiting from the mobile interface. The system can also adapt to tools being added or removed from the work environment, for example if more tools are found or a tool breaks. To do this, a user must update the number of tools on the dashboard display.

Implementation

WeBuild is implemented as a Node.js application running on a local server. Task information is stored on the server in JSON format. The application uses web sockets to communicate with each mobile phone, implemented using the socket.io module. Any device with a web browser can join by navigating to the server's IP address.

STUDY

To gain insights and an initial understanding of the impact of using WeBuild on a collaborative task, we conducted a between-subjects experiment comparing WeBuild to traditional paper manuals. Participants worked in groups of 5 on a custom designed task that combined LEGO assembly and simple woodworking. This served as an abstraction of a medium-sized task with a range of tools, materials, and required skills. We hypothesized that WeBuild would help groups complete the task more efficiently, and that it would reduce start-up time at the beginning of the task and coordination time between subtasks.

Participants

We recruited 40 participants (19 female, ages 18 - 54) from our institution and external recruitment lists. When signing up, participants were asked to rate their prior experience with LEGO and with hand tools (such as screwdrivers, wrenches, and hammers) from 1 (no experience) to 5 (very experienced). Participants were split into 8 groups of 5, with 4 groups in each condition. We balanced gender and prior experience across conditions and within each group (see Table 1 for a breakdown).

Most participants had never met, however in 4 of the 8 groups, two or three participants knew each other with varying degrees of familiarity (3 Control groups, 1 WeBuild group).

	Female	Male	LEGO Experience	Tools Experience
Control	8	12	3.2	3.45
WeBuild	11	9	3.25	3.45
Total	**19**	**21**	**3.23**	**3.45**

Table 1: Participants' gender and experience by condition.

The Task

LEGO tasks are frequently used in assembly task research studies [2, 13, 23, 33] due to their relatively clear instructions and simple assembly procedure. However, we also wanted our task to include physical tools to diversify the required skills. As such, we created a custom task that combines Funk et al.'s "pick-and-place" and "industrial assembly" task types [13]: it consists of several small LEGO models, several wooden displays of varying sizes, and a large table to hold the entire display (Figure 8). Building the wooden displays involves using a screwdriver, and building the table involves using a wrench. The table requires two people to lift it once assembled.

Figure 8: The final goal of the task used in our study.

We used the manufacturer's instructions for the LEGO models, and created our own step-by-step instructions for the table and displays. For the Control condition, the final instructions combined into a single booklet, which we have included as supplementary material for reference. For the WeBuild condition, the same instructions were entered into the system, along with the additional task information as previously described. We fixed the *task diversity* and *group priority* parameters to 0 for this study based on our knowledge of the task and the main goal of efficiency. We also disabled the "get help" feature as this task involved relatively small groups working in close quarters.

Procedure

At the start of each session, the experimenter explained the task to the group, and showed them a picture of the final goal. Participants were told to "work together to complete the task". In the Control condition, each participant was given a copy of the full instruction booklet. Participants were given a brief overview of how the booklet was organized. In the WeBuild condition, each participant was given an iPhone 5C with WeBuild loaded. The experimenter gave a brief explanation of the dashboard display, and showed participants a quick example task to explain the mobile interface. Participants in the WeBuild condition were then asked to log in, and rate their prior experience with the two skills involved in this task: LEGO and hand tools. Once all participants were ready, they were told to begin the task. When the entire group was satisfied that the task was complete, it was marked as finished. Participants then filled out a post-task questionnaire.

Measures

Our metrics were in part informed by those used in other related studies [6, 21, 33, 42].

In both conditions, we computed the overall completion time from start to finish. We also computed start-up time as the time from when the group was told to start until all five participants had begun working on something. We also computed the start and end time for each section. WeBuild saved this information directly, and in the Control condition it was determined later by watching and coding video recordings of the sessions.

From this data, we were able to compute the fraction of the total time each person spent working on sections vs. not working (i.e. waiting or coordinating with others). From the video recordings, we also computed the amount of time the group spent working in silence, to see if the amount of conversation differed between conditions.

We also measured the amount of parallel activity in each session using a similar calculation as Birnholtz et al. [6]: we split each session into 10-second time intervals and counted how many users were actively working during each interval (0-5). For each session, we averaged these scores across all intervals to determine the average amount of parallelism.

In the post-task questionnaire, participants rated the overall success of their group, as well as their group's efficiency and communication. Participants were also asked how aware they were throughout the task of their other team members' progress. Participants in the WeBuild condition also provided feedback on the system.

RESULTS

Given the scale of this study, we did not expect to formally validate our approach, however the following summary will be useful in guiding further evaluations and improvements.

Quantitative Results

For all measures, an independent t-test was used to compare the means between conditions. On average, groups in the WeBuild condition completed the task faster (23m13s) than groups in the Control condition (24m38s), however this was not significant. Figure 9 shows the progress of all eight groups over time. There appears to be more variation in the progress of Control groups. Given our small sample size such results should be considered cautiously.

The most significant observed difference between conditions was in the start-up times: WeBuild groups took an average of 24 seconds to start up, while Control groups took an average of 204 seconds ($t(6) = -5.64$, $p < 0.01$) (Figure 10a). WeBuild groups also spent a significantly higher fraction of their total time working (86%) than Control groups (72%) ($t(6) = 3.95$, $p < 0.01$) (Figure 10b). Accordingly, WeBuild groups also exhibited significantly more parallel activity (mean 4.48) than Control groups (mean 3.77) ($t(6) = 3.98$, $p < 0.01$).

On average, WeBuild groups spent more of their total time in silence (44%) than Control groups (25%), however this was not significant ($t(6) = 1.59$, $p = 0.16$) and may have also been influenced by the specific social dynamics of each group, including whether members knew each other.

Participants' answers to the post-task questions regarding success, efficiency, and communication did not differ sig-

nificantly across conditions (all had averages greater than 4.5/5). However, there was a significant difference in awareness: WeBuild participants rated themselves as *less* aware (mean 3.35/5) than Control participants (mean 4.3/5) ($t(38) = -3.45$, $p < 0.01$).

Overall, participants in the WeBuild condition responded positively to the system. The mean rating of WeBuild's overall usefulness was 4.1/5 (*SD* 0.97). Participants felt WeBuild did a good job assigning tasks (mean 4.05/5, *SD* 0.76), and that being able to step through instructions one by one was useful (mean 4.55/5, *SD* 0.69). However, ratings were mixed on the usefulness of the dashboard display (mean 2.85/5, *SD* 1.42).

Qualitative Results

Observed Challenges

Based on our observations, the Control groups seemed to have more confusion regarding which wooden pieces were for which display boards, and participants spent some time flipping back and forth in the instructions comparing different sections.

Overall, there were no major coordination issues, though in one Control session two participants ended up accidentally switching tasks, because one thought that the other had started working on their task when in fact they had not.

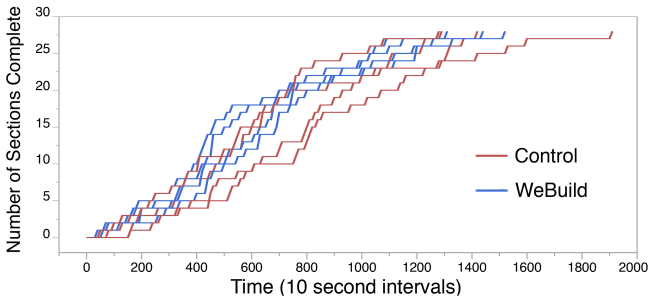


Figure 9: The progress of each group, shown as the number of sections complete in every 10-second time interval.

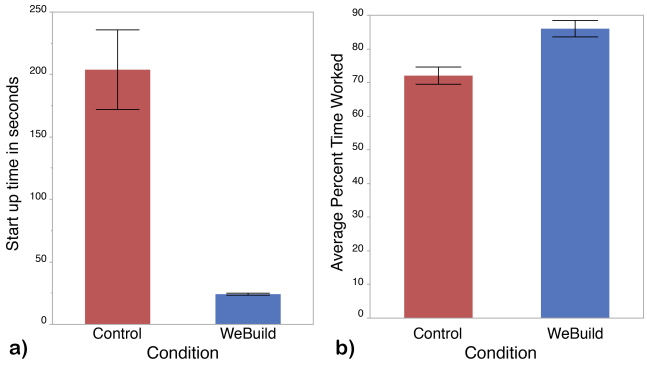


Figure 10: a) Task start-up time was significantly faster in the WeBuild condition. b) Groups in the WeBuild condition spent a larger fraction of their overall time working on the task (as opposed to waiting or coordinating). Error bars show 1 standard error from the mean.

In the WeBuild groups, participants were sometimes assigned to join sections already in progress, especially near the end of the task when there were only a few sections left. This required additional coordination as the person already working on the section had to explain what they had done to the person joining. In several cases, the first person was almost finished, and so the second person was not able to help at all.

Participant Feedback

Most participants felt their group communicated effectively. When asked how it could have been improved, participants in the Control condition suggested things like having one person be the "supervisor" to oversee group progress, and more explicit sharing of progress with each other. Several WeBuild participants answered that communication was not necessary as the system managed the task for them.

When asked how efficiency could have been improved, several Control participants mentioned that having more of an overall understanding of the task or spending more time talking through the instructions before starting would have been beneficial. WeBuild participants mentioned that sometimes having two people work on a task was unnecessary, and that when certain group members took longer than others, there was time spent waiting at the end.

Many WeBuild participants mentioned that the system was helpful as it allowed them to get started right away. As one participant stated, *"You don't need to think about the overall plan as much - you just focus on the task and assume it all works in the end"*. Suggested improvements to the system included giving participants who are waiting for a section something to do, such as help out others or provide encouragement to the group; and providing more overview information on the phone itself, as participants tended not to look away from their phones to check the dashboard.

DISCUSSION

The results from this study highlight the potential for a task distribution system to help groups coordinate physical tasks. We observed positive results overall, but it is important to note that due to the scale of our study and task, these results are suggestive rather than definitive. Our results should be treated with caution, and further formal evaluation is needed.

We predict that overall times did not differ as significantly as start-up times because once participants knew what section to do, individual section completion times were similar across conditions. Since groups spent the majority of time working on the task, the effect of the coordination improvements were most apparent at the start, but dampened across the entire session. For larger tasks where more coordination is necessary, we predict these benefits may be stronger.

Most of the non-working time spent in the WeBuild condition was near the end of the task, when participants were waiting on one or two team members to finish their final

section. This could potentially be reduced with an improved scheduling algorithm that estimates section completion times. Future implementations should also avoid assigning people to join tasks in progress unless assistance has been requested, as participants mostly found this inconvenient.

An interesting result from the study was that workers had less overall awareness in the WeBuild condition, but did not seem to mind. Participants rarely looked at the dashboard display, with most looking at it only when they had nothing else to do. This demonstrates that the system successfully eliminated any required decision-making, and workers had faith in the sections they were assigned. It would be interesting to study whether similar effects would be found for larger groups and real-world tasks where group members are likely more emotionally attached to the quality of the outcome.

The overall positive response we received from WeBuild participants was promising. It is likely that by isolating individual sections, the task distribution system reduced the need for a global understanding of the task. This allowed participants to focus on their current section without worrying about the rest, whereas Control participants had to spend time building an understanding of the overall task.

One interesting topic our work raises is that there are more dimensions to group efficiency than task completion time. For example, our system provides the option of prioritizing repetition vs. diversity, each with its own trade-offs. Essentially, this is an issue of division of labour which has a longstanding philosophical background: Marx believed that too much division and specialization would make workers less skilled overall and less motivated [28]. On the other hand, Kant believed this allowed workers to develop skills specific to their specialization and thus complete the work better [22]. We leave this choice to the users, so they can tailor the system to best fit their needs. Similarly, efficiency may not always be the main goal; for example product quality may be more important, in which case having someone rate the quality of each section could be a more useful criterion than speed.

LIMITATIONS AND FUTURE WORK

In the future we hope to test the system with more complex tasks and larger groups. For such tasks, manual coordination would likely break down as the task and group size increased. In particular, we believe our system could be adapted to large-scale efforts such as architectural construction [26] and volunteer disaster relief [27]. The main challenge of adapting to such scenarios would be creating an accurate task model.

This issue points towards one of the main areas for future improvement. The system currently requires task information and instructions to be manually entered. For each step of the instructions, an admin must enter an image of the step, information about the tools and people needed, and its dependencies. We have written a script that uses this information to segment the task into sections that can be completed independently. A simple interface for entering these inputs could aid with this process, by providing a fillable form that requests each type of information, and displays the previously entered steps for selecting dependencies. Given the rapidly growing amount of instructional content that is represented electronically, and the increasing structure in online content (e.g. instructables.com), some of this information could potentially be extracted automatically instead. In addition, prior research on generating assembly instructions automatically [2] could be extended to generate the required information for our task model.

Another limitation is that our task model assumes the task can be represented as a tree, namely that every child node has one parent. However in some tasks, several different sections could be dependent on the same child section. A more general graph representation like those used in other related work [19, 39] could handle these cases.

Regarding the distribution algorithm, it would be interesting to consider global optimizations in addition to our step-by-step algorithm. Additional prior information such as estimated completion times and more details about participants' skills could help lead to an optimal task schedule. Furthermore, our algorithm currently weighs all criteria equally when making assignments. Tweaking the weight distribution could significantly impact the outcome.

In terms of the user experience, WeBuild currently requires workers to manually advance through instructions. More adaptive support could be added by taking advantage of existing context-aware systems for automatically advancing instructions [3, 12, 34] and providing real-time error detection [17]. It would also be interesting to consider delivery of instructions through wearable devices such as smart watches [1, 26, 45] or head mounted displays [21, 38].

Our current system could also have the limitation that by reducing the amount of communication and awareness, group members find the process less enjoyable and social. Though none of our study participants mentioned this as a disadvantage, future work could consider allowing users the option to choose whom to collaborate with or which subtask to join. Other features such as the "get help" button could also be disabled as it was in our study to encourage more spontaneous collaboration.

CONCLUSION

Our initial study showed promising indications that WeBuild can help groups coordinate tasks. In addition to our suggestive results, we have contributed the design and implementation of an intelligent task distribution system that can potentially scale to larger complex tasks with an arbitrary number of workers. We also contributed a model for representing tasks and the information needed in order to make informed assignments of subtasks. We believe this research will serve as important groundwork for future efforts in coordinating collocated workers for physical tasks.

REFERENCES

1. Mario Aehnelt and Bodo Urban. 2014. Follow-Me: Smartwatch Assistance on the Shop Floor. In *HCI in Business*. Springer International Publishing, 279–287. http://doi.org/10.1007/978-3-319-07293-7_27
2. Maneesh Agrawala, Doantam Phan, Julie Heiser, et al. 2003. Designing effective step-by-step assembly instructions. *ACM SIGGRAPH 2003 Papers on - SIGGRAPH '03*, ACM Press, 828. http://doi.org/10.1145/1201775.882352
3. Stavros Antifakos, Florian Michahelles, and Bernt Schiele. 2002. Proactive Instructions for Furniture Assembly. In *UbiComp 2002: Ubiquitous Computing*. Springer Berlin Heidelberg, 351–360. http://doi.org/10.1007/3-540-45809-3_27
4. Christian Artigues. 2008. The Resource-Constrained Project Scheduling Problem. In *Resource-Constrained Project Scheduling*, Christian Artigues, Sophie Demassey and Emmanuel Nron (eds.). ISTE, London, UK, 21–36. http://doi.org/10.1002/9780470611227
5. A. Bannat, A. Bannat, F. Wallhoff, et al. 2008. Towards optimal worker assistance: A framework for adaptive selection and presentation of assembly instructions. *Proc. 1st Int. Workshop on Cognition for Technical Systems, Cotesys*.
6. Jeremy P. Birnholtz, Tovi Grossman, Clarissa Mak, and Ravin Balakrishnan. 2007. An exploratory study of input configuration and group process in a negotiation task using a large display. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*, ACM Press, 91. http://doi.org/10.1145/1240624.1240638
7. Bruno Campagnolo, Cesar A. Tacla, Emerson C. Paraiso, Gilson Y. Sato, and Milton P. Ramos. 2009. An architecture for supporting small collocated teams in cooperative software development. *2009 13th International Conference on Computer Supported Cooperative Work in Design*, IEEE, 264–269. http://doi.org/10.1109/CSCWD.2009.4968069
8. John M. Carroll, Dennis C. Neale, Philip L. Isenhour, Mary Beth Rosson, and D.Scott McCrickard. 2003. Notification and awareness: Synchronizing task-oriented collaborative activity. *International Journal of Human-Computer Studies* 58, 5: 605–632. http://doi.org/10.1016/S1071-5819(03)00024-7
9. Y.-L. Betty Chang, Stacey D. Scott, and Mark Hancock. 2014. Supporting Situation Awareness in Collaborative Tabletop Systems with Automation. *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces - ITS '14*, ACM Press, 185–194. http://doi.org/10.1145/2669485.2669496
10. Lung-Pan Cheng, Thijs Roumen, Hannes Rantzsch, et al. 2015. TurkDeck: Physical virtual reality based on people. *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15*, ACM Press, 417–426. http://doi.org/10.1145/2807442.2807463
11. Andrew Clayphan, Judy Kay, and Armin Weinberger. 2014. ScriptStorm: Scripting to enhance tabletop brainstorming. *Personal and Ubiquitous Computing* 18, 6: 1433–1453. http://doi.org/10.1007/s00779-013-0746-z
12. Mira Dontcheva, Robert R. Morris, Joel R. Brandt, et al. 2014. Combining crowdsourcing and learning to improve engagement and performance. *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, ACM Press, 3379–3388. http://doi.org/10.1145/2556288.2557217
13. Markus Funk, Thomas Kosch, Scott W. Greenwald, and Albrecht Schmidt. 2015. A benchmark for interactive augmented reality instructions for assembly tasks. *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia - MUM '15*, ACM Press, 253–257. http://doi.org/10.1145/2836041.2836067
14. Markus Funk, Thomas Kosch, and Albrecht Schmidt. 2016. Interactive worker assistance. *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '16*, ACM Press, 934–939. http://doi.org/10.1145/2971648.2971706
15. B. P. Gerkey and Maja J. Matarić. 2004. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research* 23, 9: 939–954. http://doi.org/10.1177/0278364904045564
16. D Greer and G Ruhe. 2004. Software release planning: An evolutionary and iterative approach. *Information and Software Technology* 46, 4: 243–253. http://doi.org/10.1016/j.infsof.2003.07.002
17. Ankit Gupta, Dieter Fox, Brian Curless, and Michael Cohen. 2012. DuploTrack: A realtime system for authoring and guiding Duplo block assembly. *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*, ACM Press, 389. http://doi.org/10.1145/2380116.2380167
18. Steven J. Henderson and Steven K. Feiner. 2011. Augmented reality in the psychomotor phase of a procedural task. *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, IEEE, 191–200. http://doi.org/10.1109/ISMAR.2011.6092386
19. S.J. Hu, J. Ko, L. Weyand, et al. 2011. Assembly

system design and operations for product variety. *CIRP Annals - Manufacturing Technology* 60, 2: 715–733. http://doi.org/10.1016/j.cirp.2011.05.004

20. Petra Isenberg, Danyel Fisher, Meredith Ringel Morris, Kori Inkpen, and Mary Czerwinski. 2010. An Exploratory Study of Co-located Collaborative Visual Analytics around a Tabletop Display. *Proceedings of Visual Analytics Science and Technology (VAST)*, IEEE Computer Society.
21. Steven Johnson, Madeleine Gibson, and Bilge Mutlu. 2015. Handheld or handsfree?: Remote collaboration via lightweight head-mounted displays and handheld devices. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15*, ACM Press, 1825–1836. http://doi.org/10.1145/2675133.2675176
22. Immanuel Kant. 1785. *Groundwork of the metaphysics of morals*.
23. David Kirk, Tom Rodden, and Danaë Stanton Fraser. 2007. Turn it this way: Grounding collaborative action with remote gestures. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*, ACM Press, 1039. http://doi.org/10.1145/1240624.1240782
24. Jarrod Knibbe, Tovi Grossman, and George Fitzmaurice. 2015. Smart Makerspace: An immersive instructional space for physical tasks. *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces - ITS '15*, ACM Press, 83–92. http://doi.org/10.1145/2817721.2817741
25. Robert E. Kraut, Mark D. Miller, and Jane Siegel. 1996. Collaboration in performance of physical tasks: Effects on outcomes and communication. *Proceedings of the 1996 ACM conference on Computer supported cooperative work - CSCW '96*, ACM Press, 57–66. http://doi.org/10.1145/240080.240190
26. Benjamin Lafreniere, Tovi Grossman, Fraser Anderson, et al. 2016. Crowdsourced Fabrication. *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*, ACM Press, 15–28. http://doi.org/10.1145/2984511.2984553
27. Faisal Luqman and Martin Griss. 2010. Overseer: A mobile context-aware collaboration and task management system for disaster response. *2010 Eighth International Conference on Creating, Connecting and Collaborating through Computing*, IEEE, 76–82. http://doi.org/10.1109/C5.2010.10
28. Karl Marx. 1844. Wages of Labour. In *First Manuscript, Economic and Philosophical Manuscripts*.
29. G. Michalos, S. Makris, N. Papakostas, D. Mourtzis, and G. Chryssolouris. 2010. Automotive assembly technologies review: Challenges and outlook for a flexible and adaptive approach. *CIRP Journal of Manufacturing Science and Technology* 2, 2: 81–91. http://doi.org/10.1016/j.cirpj.2009.12.001
30. Allen Newell and Paul S. Rosenbloom. 1981. Mechanisms of skill acquisition and the law of practice. In *Cognitive skills and their acquisition*, John Robert Anderson (ed.). Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1–55.
31. Linet Özdamar and Gündüz Ulusoy. 1995. A survey on the resource-constrained project scheduling problem. *IIE Transactions* 27, 5: 574–586. http://doi.org/10.1080/07408179508936773
32. David Pinelle and Carl Gutwin. 2008. Evaluating teamwork support in tabletop groupware applications using collaboration usability analysis. *Personal and Ubiquitous Computing* 12, 3: 237–254. http://doi.org/10.1007/s00779-007-0145-4
33. Abhishek Ranjan, Jeremy P. Birnholtz, and Ravin Balakrishnan. 2007. Dynamic shared visual spaces: Experimenting with automatic camera control in a remote repair task. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*, ACM Press, 1177. http://doi.org/10.1145/1240624.1240802
34. Eldon Schoop, Michelle Nguyen, Daniel Lim, Valkyrie Savage, Sean Follmer, and Björn Hartmann. 2016. Drill Sergeant: Supporting physical construction projects through an ecosystem of augmented tools. *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '16*, ACM Press, 1607–1614. http://doi.org/10.1145/2851581.2892429
35. Onn Shehory and Sarit Kraus. 1998. Methods for task allocation via agent coalition formation. *Artificial Intelligence* 101, 1: 165–200. http://doi.org/10.1016/S0004-3702(98)00045-9
36. Rajinder S. Sodhi, Brett R. Jones, David Forsyth, Brian P. Bailey, and Giuliano Maciocci. 2013. BeThere: 3D mobile collaboration with spatial input. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*, ACM Press, 179. http://doi.org/10.1145/2470654.2470679
37. Desney S. Tan, Darren Gergle, Regan Mandryk, et al. 2008. Using job-shop scheduling tasks for evaluating collocated collaboration. *Personal and Ubiquitous Computing* 12, 3: 255–267. http://doi.org/10.1007/s00779-007-0154-3
38. Arthur Tang, Charles Owen, Frank Biocca, and Weimin Mou. 2003. Comparative effectiveness of augmented reality in object assembly. *Proceedings of the conference on Human factors in computing systems - CHI '03*, ACM Press, 73.

http://doi.org/10.1145/642611.642626

39. C. Del Valle and E.F. Camacho. 1996. Automatic assembly task assignment for a multirobot environment. *Control Engineering Practice* 4, 7: 915–921. http://doi.org/10.1016/0967-0661(96)00089-5

40. Carl A. Waldspurger and William E. Weihl. 1994. Lottery scheduling: Flexible proportional-share resource management. *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation*, USENIX Association, 1.

41. James R. Wallace, Stacey D. Scott, Eugene Lai, and Deon Jajalla. 2011. Investigating the Role of a Large, Shared Display in Multi-Display Environments. *Computer Supported Cooperative Work (CSCW)* 20, 6: 529–561. http://doi.org/10.1007/s10606-011-9149-8

42. James R. Wallace, Stacey D. Scott, Taryn Stutz, Tricia Enns, and Kori Inkpen. 2009. Investigating teamwork and taskwork in single- and multi-display groupware systems. *Personal and Ubiquitous Computing* 13, 8: 569–581. http://doi.org/10.1007/s00779-009-0241-8

43. Giles Westerfield, Antonija Mitrovic, and Mark Billinghurst. 2013. Intelligent augmented reality training for assembly tasks. . Springer Berlin Heidelberg, 542–551. http://doi.org/10.1007/978-3-642-39112-5_55

44. Gwen M. Wittenbaum, Sandra I. Vaughan, and Garold Strasser. 2002. Coordination in Task-Performing Groups. In *Theory and Research on Small Groups*, R. Scott Tindale, Linda Heath, John Edwards, et al. (eds.). Kluwer Academic Publishers, Boston, 177–204. http://doi.org/10.1007/0-306-47144-2_9

45. Jens Ziegler, Sebastian Heinze, and Leon Urbas. 2015. The potential of smartwatches to support mobile industrial maintenance tasks. *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, IEEE, 1–7. http://doi.org/10.1109/ETFA.2015.7301479

46. LILLÅNGEN Wall cabinet. Retrieved from http://www.ikea.com/ca/en/catalog/products/80240789/

47. LANDON Shelf. Retrieved from https://www.jysk.ca/furniture/home-office-furniture/bookcases/landon-narrow-shelf.html

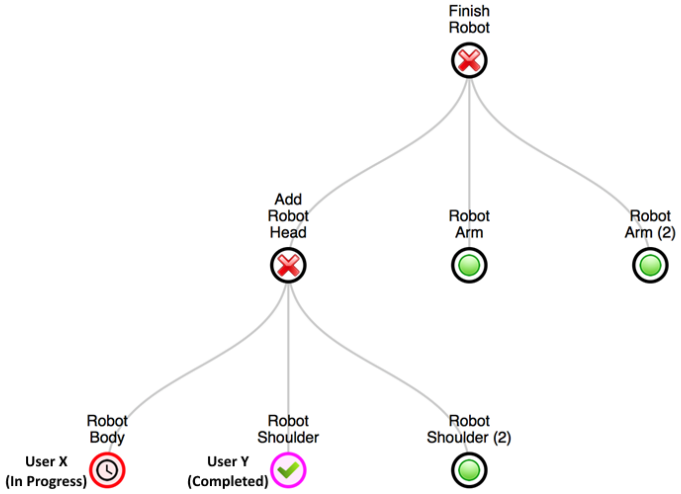
48. ME16K3000AS Over the Range Microwave. Retrieved from http://www.samsung.com/ca/consumer/home-appliances/cooking-appliances/microwave-ovens/ME16K3000AS/AC

49. Tower Bridge Set. Retrieved from http://www.meccano.com/product/p10940/tower-bridge-set

50. SEKTION High cabinet w/door & 4 drawers, white Förvara, Häggeby white. Retrieved from http://www.ikea.com/ca/en/catalog/products/S29044659/#/S89044680

APPENDIX

In this section we provide a detailed walkthrough for one iteration of our task distribution algorithm, using a simplified example task. We show the values calculated at each step of the algorithm to illustrate how the next assigned section is chosen. In this example, we assume that *task diversity* and *group priority* are both set to 0, as they were for our full study task. Assume we have two workers, X and Y. X is currently working on the Robot Body section, and Y has just finished the first Robot Shoulder section. The section assignment algorithm is triggered so that Y can be assigned a new section.



We calculate the lists U of available users and A of available sections:
$U = \{Y\}$
$A = \{$Robot Body, Robot Shoulder (2), Robot Arm, Robot Arm (2)$\}$

Note that Robot Body is included as an available section even though there is already a user working on it. This is because the maximum number of people that can work on this section (Max_i) is set to 2, so a second user could potentially join.

Now, for user Y, we build a priority list for each criterion with priority values corresponding to each section in A:

Tree Depth: $\boldsymbol{P_D} = \{3, 3, 2, 2\}$ since the first two sections (Robot Body and Robot Shoulder (2)) are at a depth of 3 in the tree, and the next two (Robot Arm and Robot Arm (2)) are at a depth of two.

Group: $\boldsymbol{P_G} = \{0, 1, 1, 1\}$ since individual work is being prioritized, and Robot Body is the only section already in progress.

Prior Experience: Assume user Y has an experience level of 3/5 with LEGO. Then $\boldsymbol{P_E} = \{3, 3, 3, 3\}$, since all sections require the LEGO skill. Since this list will be normalized later, prior experience in this case has no effect on the assignment.

Next Section: $\boldsymbol{P_N} = \{0, 0, 0, 0\}$ since the parent section of Robot Shoulder (which Y just completed) is not included in A, as it is still waiting on some other dependencies.

Repetition or diversity: Since we are prioritizing repetition in this example, we set each value $P_R[j]$ to *Similarity(j, c)* where section S_c is the one Y just completed (Robot Shoulder). Therefore, $\boldsymbol{P_R} = \{0.5, 1, 0.5, 0.5\}$ since the second section (Robot Shoulder (2)) is an exact copy of Robot Shoulder, and all other sections require the same number of tools as S_c (no tools).

Speed: For this example, speed does not affect the section assignment. This is because the only completed section so far is Robot Shoulder, and it has only been completed once by user Y. Therefore, the difference between the average time all users have taken and the average time user Y has taken will be zero (since these times are the same). Therefore, $\boldsymbol{P_S} = \{0, 0, 0, 0\}$.

To calculate the overall priority value P_{Ytotal}, we normalize each list above and add the values together element-wise:

$$\begin{aligned} P_{Ytotal} &= P_D + P_G + P_E + P_N + P_R + P_S \\ &= \{0.3, 0.3, 0.2, 0.2\} + \{0, 0.3, 0.3, 0.3\} + \{0.25, 0.25, 0.25, 0.25\} + \{0, 0, 0, 0\} + \{0.2, 0.4, 0.2, 0.2\} + \{0, 0, 0, 0\} \\ &= \{0.75, 1.25, 0.95, 0.95\} \end{aligned}$$

The highest priority value in this list is the second one (1.25), so it corresponds with the second section in A: Robot Shoulder (2). User Y is therefore assigned to Robot Shoulder (2) as the next section. This is a sensible choice given the parameters we set, since we are prioritizing for repetition and this section is identical to the one user Y just completed. We can see that its higher value came primarily from P_R, but also from P_D and P_G, since it is at the largest depth possible in the tree, and it is not already in progress and therefore will be completed individually.