28th International Conference on
Automated Planning and Scheduling

June 24-29, 2018, Delft, the Netherlands

UISP 2018

Proceedings of the 2nd Workshop on

User Interfaces and Scheduling and Planning

Edited by:

Richard G. Freedman, Jeremy D. Frank,
J Benton, Ronald P. A. Petrick

Organization

Jeremy D. Frank
NASA Ames Research Center, USA

Richard G. Freedman
University of Massachusetts Amherst, USA

J Benton
NASA Ames Research Center, USA

Ronald P. A. Petrick
Heriot-Watt University, UK

Program Committee

Amedeo Cesta
Italian National Research Council & ISTC-CNR, Italy

Tathagata Chakraborti
Arizona State University, USA

Scott Sanner
University of Toronto, Canada

Neil Yorke-Smith
Delft University of Technology, The Netherlands

Foreword

As one of the early areas of interest in artificial intelligence research, automated planning and scheduling technologies have been used in a variety of applications that involve problem solving. Although the capabilities of these technologies have matured over the years, the user interfaces of automated planners have not matured as rapidly. For those who are less familiar with the paradigms used within the ICAPS community, the advances in automated planning and scheduling are more difficult to use and interpret, even in domains where it is the most appropriate method to use. We believe that this may be one of the reasons that those who need automated planning technology are not considering its adoption, even as artificial intelligence research is finally being brought to ubiquity for the masses of average users.

Following the inaugural workshop last year, the User Interfaces and Scheduling and Planning (UISP) Workshop encourages the investigation of how user interfaces can play a role in automated planning and scheduling technologies. This is mutually beneficial because good user interfaces not only make it easier to use and develop with automated planning and scheduling technologies, but these technologies are also tools available for solving user interface problems. The research and workshop discussions last year presented ideas, questions, and challenges for what needs to be considered to not just make good user interfaces, but ones that specifically synergize with the people who will be using them with automated planning and scheduling technologies.

The works in this year's proceedings continue those trends as well as introduce new things from alternative interface modalities to examples of applications where improved interfacing can make a difference. People naturally interact via communication based on their senses of hearing and sight, and newer technologies like 'smart' devices embrace this through interaction modalities such as speech systems, virtual reality, and augmented reality. These are vastly different from the traditional computing mediums that scientists have used for years, and it is important that these evolutions are kept in mind as automated planning and scheduling technologies continue to improve. The majority of average users (and potential adopters) do not think, interpret, or communicate in computer code and mathematical representations.

Jeremy D. Frank, Richard G. Freedman, J Benton, and Ronald P. A. Petrick
June 2018

Contents

Technologies for Mixed-Initiative Plan Management for Human Space Flight

Melissa Baltrusaitis, Karen M. Feigh, and Martijn IJtsma
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology, Atlanta, GA 30332
{mbaltrusaitis3, karen.feigh, mijtsma3}@gatech.edu

Amy Pritchett
Department of Aerospace Engineering
Pennsylvania State University, State College, PA 16801
apritchett@psu.edu

William Lassiter and Martin Savelsbergh
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology, Atlanta, GA 30332
wlassiter@gatech.edu, martin.savelsbergh@isye.gatech.edu

Abstract

As humans endeavor to explore Mars and other celestial bodies further afield, we are faced with a bevy of challenges unique to deep space travel. Given that astronauts have traditionally relied on ground-based mission control to produce, manage, and adjust daily flight plans as needed, one such challenge will be the time lag in communications with mission control as a crew moves further away from the Earth. This will necessitate (automated) planning systems that will provide crews greater autonomy in managing and adapting plans to reflect the current state of the mission. This paper details the progress our research team has made in developing a mixed-initiative plan management system for use on future missions to Mars and beyond. We describe the system's design and intended capabilities and provide the results of some preliminary testing with small sample plans.

Introduction

The idea of mixed-initiative planning (MIP) (Veloso, Mulvehill, and Cox 1997; Ai-Chang et al. 2004) is to "mix" the capabilities of both automatic and human planners to generate plans that are at once compliant with a potentially large number of constraints, sensible and practical in their execution, and near-optimal with respect to one or more mission objectives. Thus a successful MIP system rests on three major building blocks: (1) a formal representation of the work encompassed in the plan (hierarchically-defined activities and resources, and the constraints put upon them); (2) the interface for a human planner to reason about (and build and modify) a plan both at the detailed level of resource scheduling and the high level abstractions, and in both nominal and off-nominal conditions; and (3) computational methods to create and, ultimately, optimize plans which can potentially scale from focused, near-term off-nominal disturbance response to more strategic planning of larger sets of activities and longer durations.

It is with these three building blocks in mind that we have developed a multi-disciplinary, integrated approach to MIP that is not tied to any one method or tool. Specifically, we are pursuing a tight integration between the needs of the human planner (as identified by specialists in cognitive engineering) and of streamlined computation (as identified by specialists in large-scale optimization) by developing a system fully capable of automated planning and re-planning but still largely reliant on human feedback and direction for plan finalization. As such, plan modifications may be triggered both automatically (according to some plan monitoring protocol) and manually (according to the needs of human planners or agents), with automatic modifications requiring approval from a human agent or planner. In order to fully accommodate both of these modification schemes, our system is comprised of three distinct components, all described briefly below, working in concert with one another, with each component intended to function as one of the three aforementioned building blocks.

Work Models that Compute (WMC): A work modeling and simulation framework that has previously been used to analyze and synthesize function allocation between air traffic controllers and an automated air traffic control system, (Pritchett, Bhattacharyya, and IJtsma 2016), a pilot and an autoflight system in the flight deck of an aircraft (Pritchett, Kim, and Feigh 2014b; 2014a), and human and robotic agents in manned space flight operations (IJtsma et al. 2017a; 2017b).

Marvin: A plan display and interface tool initially built as a timeline-tracking tool for extra-vehicular activities (EVA). It is designed to allow a mission crew to interact with a plan via both direct manipulation of individual activities and higher-level manipulation of plan priorities and constraints (Miller, Pittman, and Feigh 2017).

Optimizer: A generic term for a set of plan optimization heuristics that we will be using to modify or generate plans with the goal of reaching near-optimality with respect to one or more plan objectives. The techniques that we are currently exploring are inspired by meta-heuristic and local search concepts commonly used by operations researchers to tackle (machine) scheduling problems.

Figure 1 below is intended to represent how these three components will interact with one another in the larger system. WMC and the optimizer will be tightly coupled, communicating back and forth frequently and providing an updated plan to Marvin to allow the crew to review and finalize the modified plan.

Each component will be discussed in more detail in subsequent sections, but we first provide a brief synopsis of prior related work and a description of the plan modeling frame-

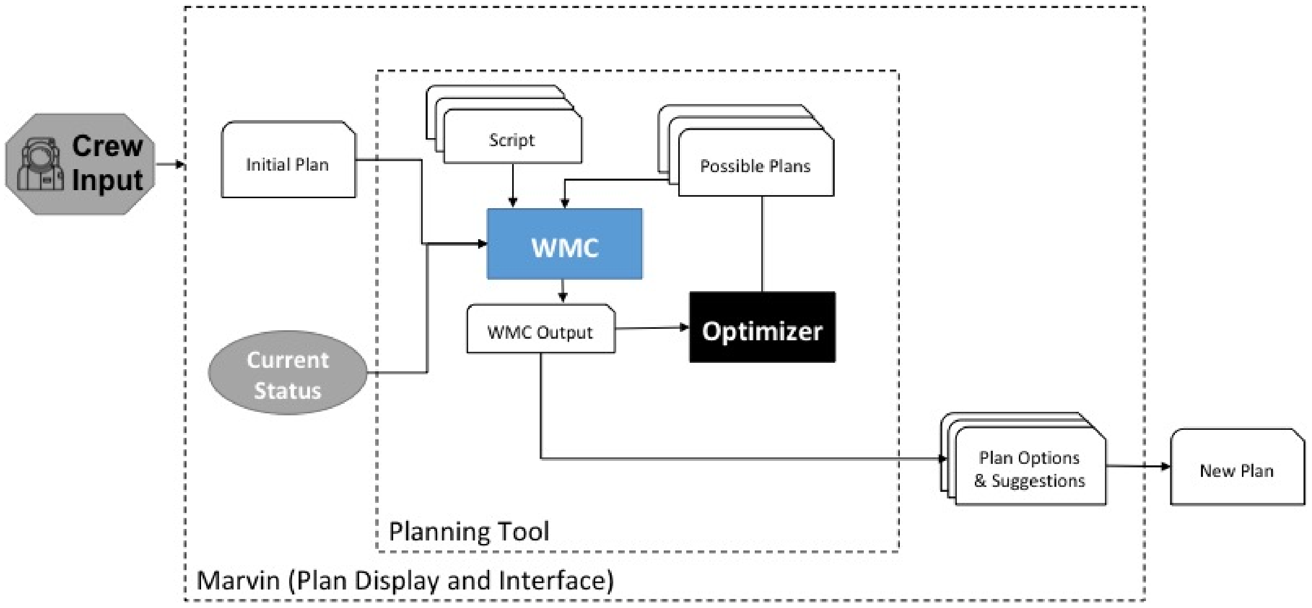


Figure 1: A visualization of our system architecture

work that we have developed for the larger system.

Related Work

Various technologies for planning have been defined in the literature. Of note, mixed-initiative planners have been studied and applied in a range of domains, including unmanned spacecraft science missions (Nothdurft et al. 2015; Biundo et al. 2011; Myers et al. 2001; Wang et al. 2013; Veloso, Mulvehill, and Cox 1997; Ai-Chang et al. 2004; Maldague et al. 1998). However, these developments have focused primarily on the technology and human aspects of MIP without examining the work of planning and the context within which it occurs. Within this limited framework, this area of the literature has debated the merits of "planning like the human" (interpreted as using some simple domain-specific heuristics which do not fully utilize the machine capabilities) (Seegebarth et al. 2012; van Wezel and Jorna 2009) or "using AI to plan" (interpreted as automatically generating plans which the human may or may not be able to interact with and modify, yet are reported to need intervention some 30-40% of the time, particularly when the system does not have full knowledge of the goals or constraints) (Ai-Chang et al. 2004; Cegarra and van Wezel 2012). In many of the "using AI to plan" cases, the underlying representation of the plan's rationale has not been clear to the human, a difficulty reflected in the costs of user interfaces associated with these various systems, reported as being up to 70% of the cost of development (Cegarra and van Wezel 2012).

Critiques of MIP systems not developed around explicit analysis of the underlying "work" to be performed have noted key obstacles to their implementation. For example, MIP has historically separated the "planning" of activities from the "scheduling" of resources; in contrast, the work of space flight planning must simultaneously consider what is feasible in terms of resource constraints when planning activities (Myers et al. 2001; Maldague et al. 1998). Likewise, human planners were found to use in their work non-exclusive decompositions reflecting activities operating upon shared resources, and to create annotations within structures intended to define resources to instead reflect more abstract information about the plan. From a cognitive engineering perspective, the obstacles faced by the current state of the art reflect a lack of emphasis on understanding the work of planning.

An awareness of the context of planning, i.e. recognizing that there can be multiple different strategies for planning based on the scenario within which planning is taking place, is equally critical to building an effective MIP system. For example, the notion of cognitive control describes how human planners may employ different strategies depending on the balance of resources (time-available, relevant knowledge, etc.) versus demands (time pressure, unfamiliar situations, etc.) (Hollnagel 1993). Feigh's work, for example, has demonstrated how different strategies for airline operations planning can be supported by different interfaces in such a way that simpler, more-efficient-but-less-optimal strategies can be easily invoked when efficiency is to be valued and a different interface can be toggled to when time allows for more strategic approaches (Feigh and Pritchett 2010).

Rather than relying on AI-based planning algorithms, we are employing methods developed in operations research (OR), a field that has long examined problems in planning and scheduling, that have been extremely successful in solving large, complex, practical optimization problems. While OR began as the study of deterministic and single-objective (and thus largely impractical) problems, focus in recent years has shifted to fields with more practical applications—

stochastic and robust optimization (Kleywegt and Shapiro 2007; Bertsimas, Brown, and Caramanis 2011), with which plans that explicitly account for quantifiable future uncertainties can be created, online optimization (Sgall 1998; Kalai and Vempala 2005), with which plans can be adjusted in real-time based on newly revealed information, and multi-objective optimization (Ehrgott 2010), with which trade-offs between competing plan objectives may be explored and quantified. Local search techniques and heuristics have also received quite a bit of attention and have proven to be quite useful in solving large scheduling problems (Aarts and Lenstra 2003). To our knowledge, these more advanced OR techniques have never been integrated into a single decision framework akin to the one necessary for deep space exploration missions.

Plan-Modeling Framework

A robust plan-modeling framework is crucial to the efficacy of any MIP system, and especially so for our system given that it requires communication across three separate platforms. The common framework that we have developed is based on three main high-level constructs: activities, agents, and resources. All members of these three constructs have their own set of attributes that we have divided into three separate categories: characteristics, current state, and plan information. In the operation of our system, characteristics are static input data, current state is dynamic input data, and plan information is output data. A more detailed discussion of each construct and its relevant attributes is given below.

Activities

Activities comprise everything that must be done in a given time frame and are organized in a four-tier structure, with activities (highest level) subdivided into tasks, tasks into subtasks, and subtasks into procedures. This organizational structure helps to facilitate flexibility with respect to the amount of granularity required for planning or re-planning. In the context of planning, we have developed and categorized relevant attributes of activities as follows:

Characteristics Relevant activity characteristics include

- A unique identifier (e.g. activity name)
- Earliest (Latest) allowed start (completion) time
- Location and duration
- Activity tier, and parent (one tier higher) and child (one tier lower) activities, if applicable
- Any preceding or succeeding activities
- Difficulty level (subjective, rated on to a numeric scale)
- Agent skills and/or resources required for execution.

Current State The only current state information needed for an activity is a flag indicating whether or not it is already included in the current plan.

Plan Information Plan information for an activity includes the agent(s) performing the activity, its start and end times, and the identifiers of any resources it is using.

Agents

An agent is any entity capable of performing work; a plan's list of agents may include human crew members, robonauts, robotic arms, astrobees and other such entities. The attributes of agents that we have deemed relevant in the context of planning are as follows:

Characteristic Relevant agent characteristics include a unique identifier (e.g. agent name), oxygen and/or energy consumption rates, agent skills (in order to match certain agents with certain activities), and movement speed/range of motion.

Current State Current state information for agents includes location, resources in possession (e.g. tools for completing a maintenance activity), agent fatigue level (calculated according to difficulty levels of activities executed) and times available (times in the plan when the agent is not scheduled to be working).

Plan Information Plan information for a particular agent is simply the set of activities that the agent has been tasked with executing in the plan.

Resources

A resource is an inanimate object or supply which must be utilized to perform certain activities. A resource may be a tool, replacement part, EVA suit, oxygen tank, power cell, or other such implement. Relevant attributes of resources in the context of planning are as follows:

Characteristic Relevant resource characteristics include a unique identifier (e.g. resource name + number if there are more than one of something), energy consumption rate, maximum battery and/or oxygen capacity (when applicable), and storage location.

Current State Current state information for resources includes location, times available, and oxygen and battery levels when applicable.

Plan Information Plan information for a particular resource is, similar to that of an agent, simply the set of activities that utilize the resource in the plan.

These three constructs and their relevant attributes form the backbone of our planning structure and must be interpreted and utilized by each component of our system architecture. We now proceed to discussing this architecture in greater detail.

System Architecture

As was mentioned in the introduction, we have designed our MIP system to address the needs of both the human planner and streamlined computation. Of vital importance to the human planner is an adequate representation of the work to be completed and a conception of how this work will flow in a given plan. Streamlined computation requires a structure that lends itself to the creation of robust, efficient modification and optimization algorithms. With these ideas in mind, this section is dedicated to our efforts so far in (1) the modeling and simulation of work and (2) the development of plan optimization techniques.

Modeling Work

Provided the emphasis on the human in MIP, we have extended this influence into the development of our model's work constraints. By taking a human-centered approach, we have not only considered traditional technological constraints, but we have also examined cognitive, social, and physical factors that influence work. Thus, we have defined the following requirements by resource, temporal, and agent activities.

Resource Constraints

- Activity X requires Y resource <oxygen, CO2, water, power, thermal> amount
- No more than Y resources used in Z time period

Temporal Constraints

- Activity X <before, after, at same time> Activity Y
- Perform Activity X at Time T
- Perform Activity X at RelativeTime R <prior, after> to Time T
- Perform Activity X at RelativeTime R <prior, after> to Activity Y
- Combine Activity X and Y into Activity Z in this order
- Break Activity X into Activity Y and Z at Step J
- Add task X <now, Time T>
- Remove Activity X

Agent Constraints

- Assign Activity X to Agent Y
- Split Activity X across Agents Y and Z

To further examine the factors that influence work in a MIP system, specifically astronaut-system interactions, we developed three scenarios that span varying levels of safety and time considerations. These scenarios include emergency, mission impact, and optional events, with each defined in detail in Figure 2 above. Building on these scenarios by creating storyboards, information flow diagrams, and initial user environment design documents (Beyer and Holtzblatt 1998), we reinforced the proposed work model for the optimization algorithm and provided a foundation for initial interface design requirements.

Simulating Work: WMC

WMC is a computational simulation framework that has been developed over several years, originally as a means of evaluating function allocations between various automation, robotic and human agents (Pritchett, Kim, and Feigh 2014b; 2014a). The high-level model structure of the framework consists of a work model and an agent model, which are then simulated through time as agents performing the work in the work model. The work model is a representation of the activities of a plan as well as the resources that are required for the work. Each activity has attributes that define constraints such as its location, the resources that are used or consumed and the duration it has. An agent model consists of simple heuristics for executing each activity. WMC takes two inputs: a function allocation that assigns each activity to a performing agent and a plan that consists of steps with activities and scheduled times.

The simulation loads the plan and function allocation into its simulation core, and steps through this internal activity list sorted by scheduled time and calls associated agent models to perform the activities due at the current simulation time. When the agent model is called by the simulation's core to perform an activity, its internal checks account for constraints in the work and the agent, such as the location and availability of the required resources, the maximum number of activities each agent can perform at a time, and the location of each agent.

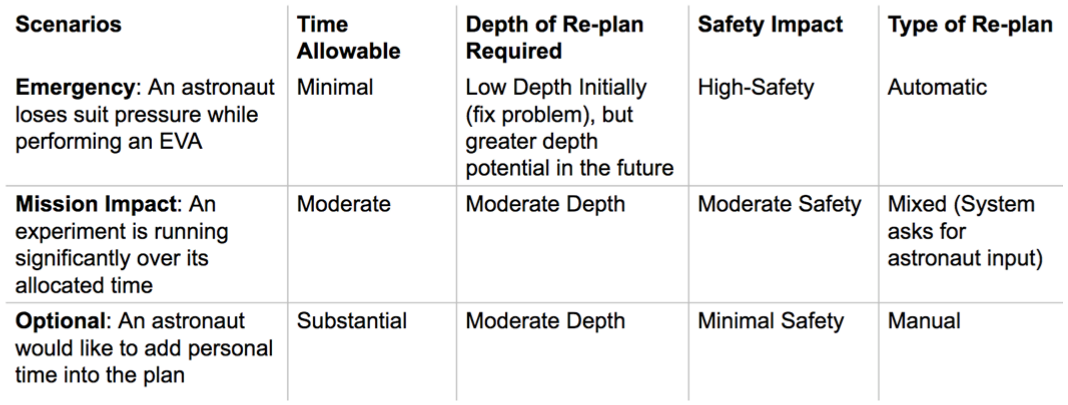
When constraints are not met, the agent model has simple heuristics to resolve a constraint violation. For example, when a resource for an activity is not available at its original scheduled time, the agent model will delay the activity until this resource becomes available (when the activity that was occupying the resource finishes). In case locations of resources or agents do not match, the simulation can account for required traversal times to fetch resources or change agent locations. These alterations to the plan are logged and fed back to the optimization algorithm for further iterations.

The framework furthermore logs several metrics during simulations that can subsequently be used as objectives in the optimization process. Examples of performance metrics are the total time to completion, the idle or down time of agents and the total time on task for each agent. Additionally, WMC logs metrics capture some of the coordination or teamwork that is required between agents to make a plan work. For example, instances of information sharing between two agents (based on their activities and the respective timing) are logged as a measure for the required communication. When resources need to be shared between agents, WMC logs the transfer of these resources as requirements for physical interaction between agents in which the resources are handed over. The traversal time associated with such resource management, or simply to move from one location to the other between consecutive activities, is also logged as a metric.

Optimization

The plan-modeling framework detailed in the previous section lends itself nicely to a scheduling theory-based approach to modification and optimization. Scheduling theory is a branch of OR that has grown out of efforts to solve variations of the classic machine scheduling problem: the problem of scheduling n "jobs" on m "machines" in such a way that some objective (e.g., the completion time of the last job) is optimized. In the context of automated planning for human space flight, activities can be thought of as "jobs" and agents as "machines". Many of the temporal and agent constraints discussed above have natural representations within the confines of a machine scheduling problem, and resource constraints can fairly easily be tracked at a high level as well.

Thus, the Optimizer briefly described earlier essentially



Scenarios	**Time Allowable**	**Depth of Re-plan Required**	**Safety Impact**	**Type of Re-plan**
Emergency: An astronaut loses suit pressure while performing an EVA	Minimal	Low Depth Initially (fix problem), but greater depth potential in the future	High-Safety	Automatic
Mission Impact: An experiment is running significantly over its allocated time	Moderate	Moderate Depth	Moderate Safety	Mixed (System asks for astronaut input)
Optional: An astronaut would like to add personal time into the plan	Substantial	Moderate Depth	Minimal Safety	Manual

Figure 2: Detailed description of the three scenarios. In this table, we consider: the amount of time that the optimizing system has to evaluate and present a new plan to the team – time allowable; the amount of the plan that will need to be evaluated and potentially modified – depth of re-plan; the introduction of new constraints that prioritize crew and structure health – safety impact; and the type of method used to trigger a re-planning event — type of re-plan.

endeavors to solve (or come very close to solving) a machine scheduling problem over given sets of activities and agents. Given that the problem of optimally scheduling jobs on two or more machines with an objective as simple as time to completion has been shown to be $\mathcal{NP}$-complete (i.e., computationally intractable for large instances) (Ullman 1975), solution techniques are generally heuristic in nature. Local (or neighborhood) search is one such solution technique that has proven effective in handling large instances, especially when embedded in a meta-heuristic framework, making it our method of choice for use in the Optimizer.

Our current local search algorithm is relatively simple. The "neighborhood" around a given schedule is defined to be all schedules that can be obtained from it by removing a single activity and reinserting it elsewhere. An objective value to optimize is specified at the outset, and at each iteration a schedule in the neighborhood is selected at random. If this new schedule has an objective value that is at least as good as the current schedule, the algorithm "jumps" from the latter to the former, and then repeats. Otherwise, the current schedule remains as is and a different schedule in the neighborhood is selected in the next iteration. This process continues either for a set number of iterations or until the schedule's objective value reaches a threshold, depending on the user's preference. The algorithm can also consider multiple objectives via added constraints that prevent jumping to a new schedule if doing so would cause one or more secondary objective values to exceed specified thresholds.

Given that as many as several thousand or more such jumps may be necessary to obtain a close-to-optimal schedule, examining resource availability, usage, and consumption in depth at each iteration has the potential to be computationally prohibitive. With this in mind, our algorithm treats resource-related constraints at a coarse, high level and relies on WMCs detailed evaluation process for resource-related feedback and suggested schedule changes. The section below discusses the integration between the Optimizer and WMC in greater depth.

Integration

The optimization algorithm and WMC are written as separate processes in C++. Both contain similar objects, albeit for different purposes: the optimization algorithm uses heuristics to reason using simple constraints and to construct new plans, and WMC uses the objects to evaluate a plan using a range of metrics. However, to assure the objects in the two processes operate consistently on the same data, the class attributes of each objects are populated from the same XML input file, as shown in Figure 3.

This XML file defines the activities of the plan, the available agents and resources, and provides the optimizer and WMC with information on the resource, temporal, and agent constraints. Any input from the astronaut through the Marvin interface will also be defined in this XML input format. The file follows the same basic format as described in the Modeling Framework section and thus contains static information on objects (the Characteristics attributes), an initial state of dynamic attributes (Current State attributes), and the original plan that is used as a starting point for the optimization (Plan Information attributes).

Then, during the optimization process, as different plans are constructed and evaluated in WMC, the dynamic attributes of objects change. To exchange this changing information - a plan change to be simulated in WMC and the corresponding metrics to be considered in the optimization algorithm - we use a C++ *shared memory object*. Compared to other methods like external log files or message passing, shared memory is considered the most computationally efficient method for interchanging data between processes. To reduce the required communication, only the dynamic attributes of objects are communicated through the shared memory, including:

- Scheduled time for each activity.
- The function allocation denoting which agent is assigned to perform each activity.
- Metrics of interest for the plan. Metrics in the initial im-

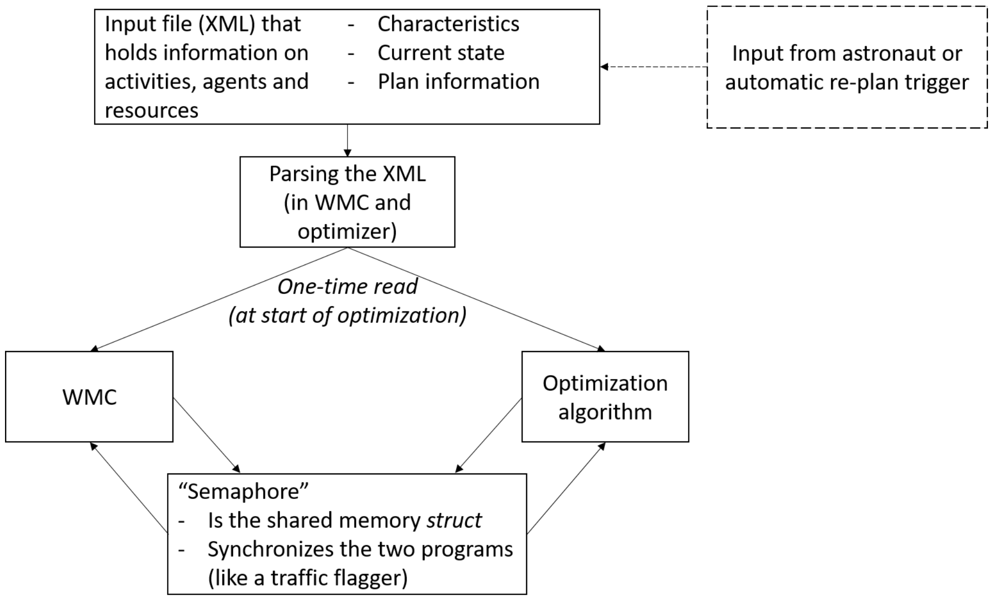

Input file (XML) that holds information on activities, agents and resources
- Characteristics
- Current state
- Plan information
Input from astronaut or automatic re-plan trigger
Parsing the XML (in WMC and optimizer)
One-time read (at start of optimization)
WMC
Optimization algorithm
"Semaphore"
- Is the shared memory struct
- Synchronizes the two programs (like a traffic flagger)

Figure 3: Diagram of the integration between the optimization algorithm and the WMC simulation framework.

plementation include the makespan (i.e., the total duration of a schedule), mental and physical workload estimates for each agent, time on task for each agent, the total number of physical resource exchanges between agents, and a list of actions that are performed past their deadline.

Sample Case Study

To validate the coupling of the optimization algorithm and WMC, we have constructed a use case emulating the re-planning of one day's existing schedule to produce a feasible, modified solution. To enable this process, we have developed an example 12-hour mission schedule that reflects the work, exercise, and leisure standards that are currently employed by the Flight Planning team at NASA JSC. The names of the activities in the schedule are shown in Figure 4. Each activity in the schedule contains inherent characteristics, including location, skills, mental and physical difficulty, and temporal and resource dependencies, which are used to constrain new solutions. The activity constraints contained within the case study are designed explicitly to ensure that the schedule captures all of the plan-specific work requirements that were outlined in the Modeling Work section.

For this particular case study we have limited ourselves to considering two distinct objectives: (1) the plan's makespan (i.e. overall time to completion) and (2) workload balance among agents. The results displayed below were obtained by optimizing with respect to makespan, but the plan can be optimized with respect to workload balance as well by aggregating the difficulty levels of assigned activities across all agents and minimizing variance between agents.

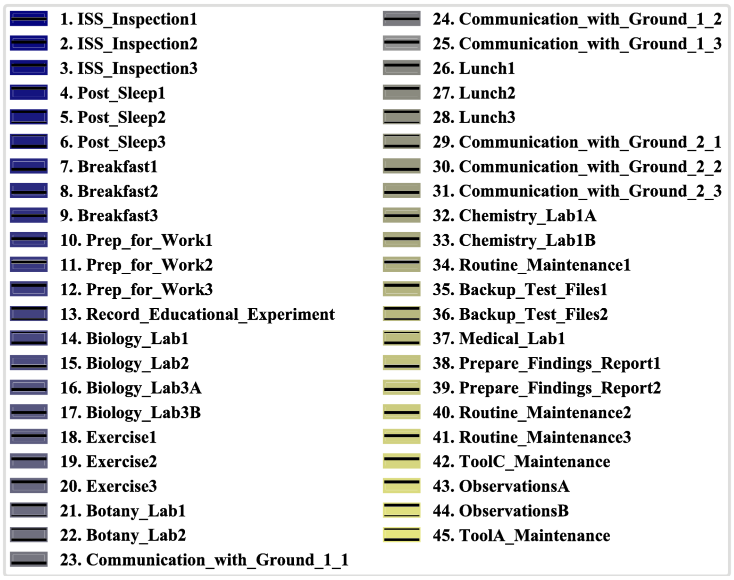
Figure 5 shows the existing 12-hour mission schedule before optimization (subfigure a) and the schedule after optimization (subfigure b), in the form of time traces of when each activity is performed and by which agent. For this preliminary case study, we let the optimization algorithm perform ten thousand iterations, and subsequently simulated the resulting best option in WMC for more detailed evaluation.

The optimization process resulted in a plan with a 45-minute reduction in makespan, following a considerable amount of activity re-scheduling (e.g., the order of activities for Agent 3 has changed notably) and/or re-assignment (e.g., activity 14 is re-assigned from Agent 1 to Agent 2). All changes that involve re-assigning activities from one agent to another have been highlighted using dashed lines; most re-assignments are found between Agents 1 and 2. Agent 3 has mostly changes in the order of his/her activities. These changes result in a more condensed schedule, with a more balanced workload distribution. Finally, the makespan cannot be reduced any further since activities 38 and 39 ("Prepare Findings Report 1" and "Prepare Findings Report 2") both have constraints that require them to be executed at their respective times.

Future Work

In this paper, we present a novel approach to MIP for long-duration spaceflight that focuses on the tight coupling of three components: a work-modeling framework, an optimization algorithm, and an intuitive user interface. While the case study proved the feasibility of this design, significant work remains to present a formal solution.

Better modeling the true nature of work is one focus area. We are continuing to refine the example schedule to better reflect mission-planning constraints and manage human abilities and preferences, including reducing the spacing between activities. Additionally, we are modeling a new mission-impact scenario that is focused on the inclusion of an unplanned maintenance task, which will require direct input from ground control.



1. ISS_Inspection1
2. ISS_Inspection2
3. ISS_Inspection3
4. Post_Sleep1
5. Post_Sleep2
6. Post_Sleep3
7. Breakfast1
8. Breakfast2
9. Breakfast3
10. Prep_for_Work1
11. Prep_for_Work2
12. Prep_for_Work3
13. Record_Educational_Experiment
14. Biology_Lab1
15. Biology_Lab2
16. Biology_Lab3A
17. Biology_Lab3B
18. Exercise1
19. Exercise2
20. Exercise3
21. Botany_Lab1
22. Botany_Lab2
23. Communication_with_Ground_1_1
24. Communication_with_Ground_1_2
25. Communication_with_Ground_1_3
26. Lunch1
27. Lunch2
28. Lunch3
29. Communication_with_Ground_2_1
30. Communication_with_Ground_2_2
31. Communication_with_Ground_2_3
32. Chemistry_Lab1A
33. Chemistry_Lab1B
34. Routine_Maintenance1
35. Backup_Test_Files1
36. Backup_Test_Files2
37. Medical_Lab1
38. Prepare_Findings_Report1
39. Prepare_Findings_Report2
40. Routine_Maintenance2
41. Routine_Maintenance3
42. ToolC_Maintenance
43. ObservationsA
44. ObservationsB
45. ToolA_Maintenance

Figure 4: Activities to be completed as part of case study.

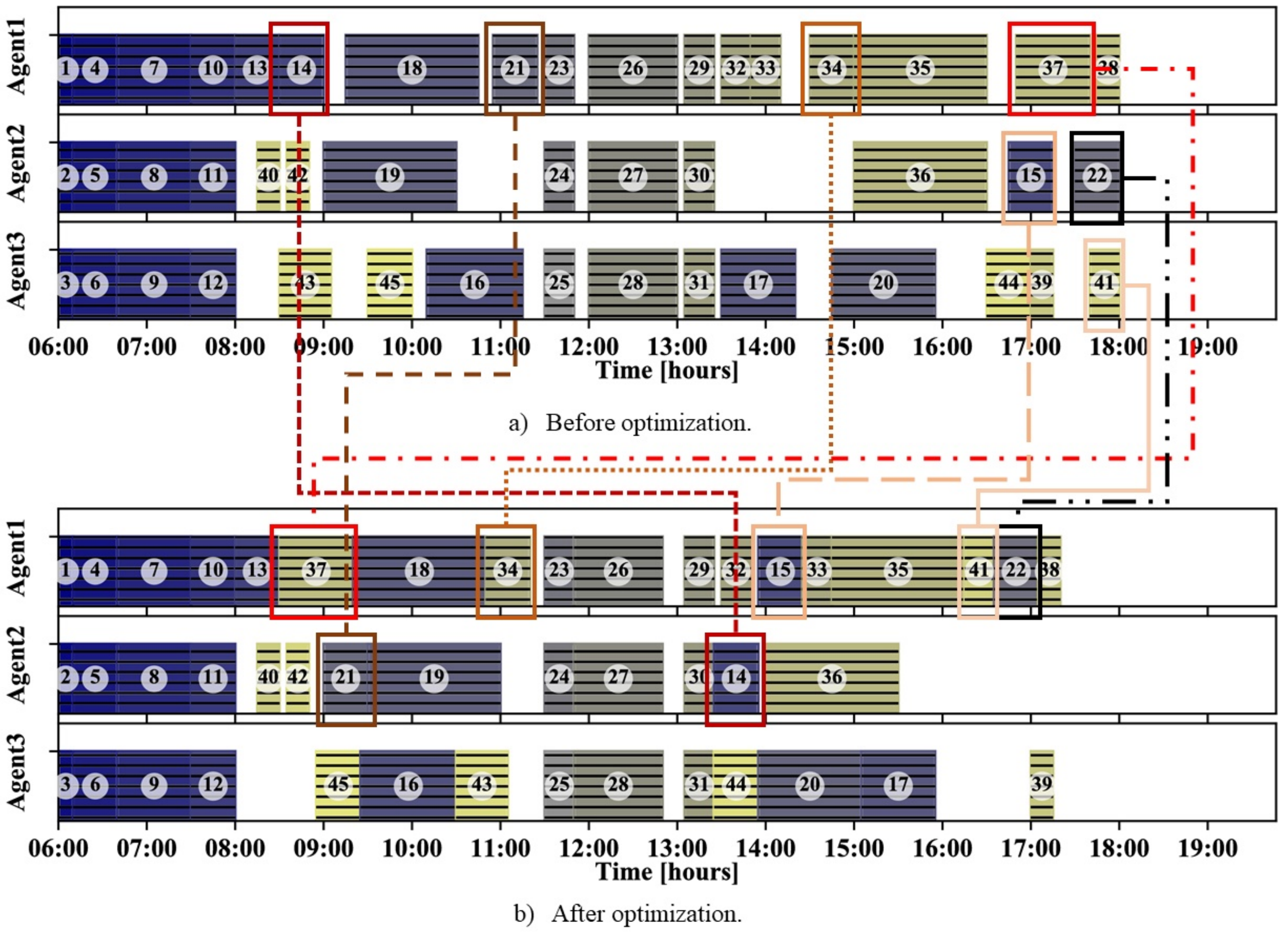


Figure 5: Time trace of existing mission schedule before and after optimization.

Future work on the WMC framework will include feeding back to the Optimizer more advanced metrics that would not be easily evaluated in the optimization algorithm, such as the required traversal of astronauts through the space station and more detailed resource-related metrics. Another possible addition to WMC is to simulate uncertainty in the action duration, allowing for estimates of the robustness of a plan to natural variation in the work.

Work will also continue on improving our local search heuristic for optimization. We anticipate developing more efficient and problem-tailored algorithms through the use of metaheuristics such as tabu search, adaptive large neighborhood search, and multi-start search, as well as implementing techniques to address new, more subtle objectives such as agent preferences, resource consumption, and total number of physical resource exchanges between agents.

We also plan to continue expanding upon the integration between the optimization algorithm and WMC, writing more metrics from WMC to the shared memory and having the optimizer use these simulation results in clever ways in follow-up iterations. One important aspect to consider in the integration is that when WMC performs its more detailed evaluation it should provide rich feedback to the optimizer, i.e., not just the metric values but also reasoning behind any delays or changes in the schedule. Likewise, when the WMC and optimizer combination is integrated with a human interface, similar kind of informative feedback should be fed back to the system's user.

Finally, the development of the interface is an intended area for future work. To date, we have developed scenarios, storyboards, information flows, and user environment designs to derive initial interface design requirements. Thus, the next step will be to convert these requirements into mockups and prototypes, and ultimately, a functional and intuitive application.

Acknowledgments

This work is funded as part of a NASA Early Stage Innovation Grant (NNX17AD12G), with Steve Hillenius and Jeremy Frank serving as Program Managers. This work is solely the responsibility of its authors and does not necessarily represent the official views of NASA.

References

Aarts, E. H. L., and Lenstra, J. K. 2003. *Local Search in Combinatorial Optimization*. Princeton University Press.

Ai-Chang, M.; Bresina, J.; Charest, L.; Chase, A.; Hsu, J. C. J.; Jonsson, A.; Kanefsky, B.; Morris, P.; Rajan, K.; Yglesias, J.; Chafin, B. G.; Dias, W. C.; and Maldague, P. F. 2004. Mapgen: mixed-initiative planning and scheduling for the mars exploration rover mission. *IEEE Intelligent Systems* 19(1):8–12.

Bertsimas, D.; Brown, D. B.; and Caramanis, C. 2011. Theory and applications of robust optimization. *SIAM Review* 53(3):464–501.

Beyer, H., and Holtzblatt, K. 1998. *Contextual Design*. San Diego, CA: Academic Press.

Biundo, S.; Bercher, P.; Geier, T.; Mller, F.; and Schattenberg, B. 2011. Advanced user assistance based on ai planning. *Cognitive Systems Research* 12(3):219 – 236. Special Issue on Complex Cognition.

Cegarra, J., and van Wezel, W. 2012. Revisiting decision support systems for cognitive readiness: A contribution to unstructured and complex scheduling situations. *Journal of Cognitive Engineering and Decision Making* 6(3):299–324.

Ehrgott, M. 2010. *Multicriteria optimization*. Springer.

Feigh, K., and Pritchett, A. 2010. Modeling work for cognitive work support system design in operational control centers. *Journal of Cognitive Engineering and Decision Making* 4:126.

Hollnagel, E. 1993. *Human Reliability Analysis: Context and Control*. Academic Press.

IJtsma, M.; Ma, L. M.; Pritchett, A. R.; and Feigh, K. M. 2017a. Work Dynamics of Taskwork and Teamwork in Function Allocation for Manned Spaceflight Operations. In *International Symposium on Aviation Psychology*, 554–559.

IJtsma, M.; Pritchett, A. R.; Ma, L. M.; and Feigh, K. M. 2017b. Modeling Human-Robot Interaction to Inform Function Allocation in Manned Spaceflight Operations. In *Robotics: Science and Systems, Workshop: Bridging the Gap in Space Robotics*.

Kalai, A., and Vempala, S. 2005. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences* 71(3):291 – 307. Learning Theory 2003.

Kleywegt, A. J., and Shapiro, A. 2007. *Stochastic Optimization*. Wiley-Blackwell. chapter 102, 2625–2649.

Maldague, P.; Ko, A.; Page, D.; and Starbird, T. 1998. Apgen: A multi-mission semi-automated planning tool. In *First International NASA Workshop on Planning and Scheduling*.

Miller, M. J.; Pittman, C. P.; and Feigh, K. M. 2017. Next-generation human extravehicular spaceflight operations support systems development. In *International Aeronautical Congress (IAC)*.

Myers, K.; Smith, S.; Hildum, D. W.; Jarvis, P.; and de Lacaze, R. 2001. Integrating planning and scheduling through adaptation of resource intensity estimates. In *Proceedings 5th European Conference on Planning*.

Nothdurft, F.; Behnke, G.; Bercher, P.; Biundo, S.; and Minker, W. 2015. The interplay of user-centered dialog systems and ai planning. *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Pritchett, A. R.; Bhattacharyya, R. P.; and IJtsma, M. 2016. Computational assessment of authority and responsibility in air traffic concepts of operation. *Journal of Air Transportation* 24(3):93–102.

Pritchett, A. R.; Kim, S. Y.; and Feigh, K. M. 2014a. Measuring human-automation function allocation. *Journal of Cognitive Engineering and Decision Making* 8(1):52–77.

Pritchett, A. R.; Kim, S. Y.; and Feigh, K. M. 2014b. Modeling human-automation function allocation. *Journal of Cognitive Engineering and Decision Making* 8(1):33–51.

Seegebarth, B.; Muller, F.; Schattenberg, B.; and Biundo, S. 2012. Making hybrid plans more clear to human users-a formal approach for generating sound explanations. In *Twenty-Second International Conference on Automated Planning and Scheduling*.

Sgall, J. 1998. *On-line scheduling*. Berlin, Heidelberg: Springer Berlin Heidelberg. 196–231.

Ullman, J. 1975. Np-complete scheduling problems. *Journal of Computer and System Sciences* 10(3):384 – 393.

van Wezel, W., and Jorna, R. 2009. Cognition, tasks and planning: supporting the planning of shunting operations at the netherlands railways. *Cognition, Technology & Work* 11(2):165–176.

Veloso, M. M.; Mulvehill, A. M.; and Cox, M. T. 1997. Rationale-supported mixed-initiative case-based planning. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, 1072–1077. AAAI Press.

Wang, Z.; Wang, H.-W.; Qi, C.; and Wang, J. 2013. A resource enhanced htn planning approach for emergency decision-making. *Applied Intelligence* 38(2):226–238.

Visualizations for an Explainable Planning Agent

Tathagata Chakraborti[1] and **Kshitij P. Fadnis**[2] and **Kartik Talamadupula**[2] and **Mishal Dholakia**[2]
Biplav Srivastava[2] and **Jeffrey O. Kephart**[2] and **Rachel K. E. Bellamy**[2]

[1] Computer Science Department, Arizona State University, Tempe, AZ 85281 USA
tchakra2 @ asu.edu

[2]IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA
{ kpfadnis, krtalamad, mdholak, biplavs, kephart, rachel } @ us.ibm.com

Abstract

In this paper, we report on the visualization capabilities of an Explainable AI Planning (XAIP) agent that can support human in the loop decision making. Imposing transparency and explainability requirements on such agents is especially important in order to establish trust and common ground with the end-to-end automated planning system. Visualizing the agent's internal decision making processes is a crucial step towards achieving this. This may include externalizing the "brain" of the agent – starting from its sensory inputs, to progressively higher order decisions made by it in order to drive its planning components. We also show how the planner can bootstrap on the latest techniques in explainable planning to cast plan visualization as a plan explanation problem, and thus provide concise model based visualization of its plans. We demonstrate these functionalities in the context of the automated planning components of a smart assistant in an instrumented meeting space.

Introduction

Advancements in the fields of speech, language, and search have led to ubiquitous personalized assistants like the Amazon Echo, Google Home, Apple Siri, etc. Even though these assistants have mastered a narrow category of interaction in specific domains, they mostly operate in passive mode – i.e. they merely respond via a set of predefined scripts, most of which are written to specification. In order to evolve towards truly smart assistants, the need for (pro)active collaboration and decision support capabilities is paramount. Automated planning offer a promising alternative to this drudgery of repetitive and scripted interaction. The use of planners allows automated assistants to be imbued with the complementary capabilities of being nimble and proactive on the one hand, while still allowing specific knowledge to be coded in the form of domain models. Additionally, planning algorithms have long excelled (Myers 1996; Sengupta et al. 2017) in the presence of humans in the loop for complex collaborative decision making tasks.

eXplainable AI Planning (XAIP) While planners have always adapted to accept various kinds of inputs from humans, only recently has there been a concerted effort on the other side of the problem: making the outputs of the planning process more palatable to human decision makers. The paradigm of eXplainable AI Planning (XAIP) (Fox, Long, and Magazzeni 2017) has become a central theme around which much of this research has coalesced. In this paradigm, emphasis is laid on the qualities of *trust*, *interaction*, and *transparency* that an AI system is endowed with. The key contributions to explainability are the resolution of critical exploratory questions – why did the system do something a particular way, why did it not do some other thing, why was its decision optimal, and why the evolving world may force the system to replan.

Role of Visualization in XAIP One of the keys towards achieving an XAIP agent is visualization. The planning community has recently made a concerted effort to support the visualization of key components of the end-to-end planning process: from the modeling of domains (Bryce et al. 2017); to assisting with plan management (Izygon, Kortenkamp, and Molin 2008); and beyond (Sengupta et al. 2017; Benton et al. 2017). For an end-to-end planning system, this becomes even more challenging since the systems state is determined by information at different levels of abstraction which are being coalesced in the course of decision making. A recent workshop (Freedman and Frank 2017) outlines these challenges in a call to arms to the community on the topic of visualization and XAIP.

Contribution It is in this spirit that we present a set of visualization capabilities for an XAIP agent that assists with human in the loop decision making tasks: specifically in the case of this paper, assistance in an instrumented meeting space. We introduce the end-to-end planning agent, `Mr.Jones` (Chakraborti et al. 2017b), and the visualizations that we endow it with. We then provide fielded demonstrations of the visualizations, and describe the details that lie under the hood of these capabilities.

Introducing `Mr.Jones`

First, we introduce `Mr.Jones` (Chakraborti et al. 2017b), situated in the `CEL` – the Cognitive Environments Laboratory – at IBM's T.J. Watson Research Center. `Mr.Jones` is designed to embody the key properties of a proactive assistant while fulfilling the properties desired of an XAIP agent.

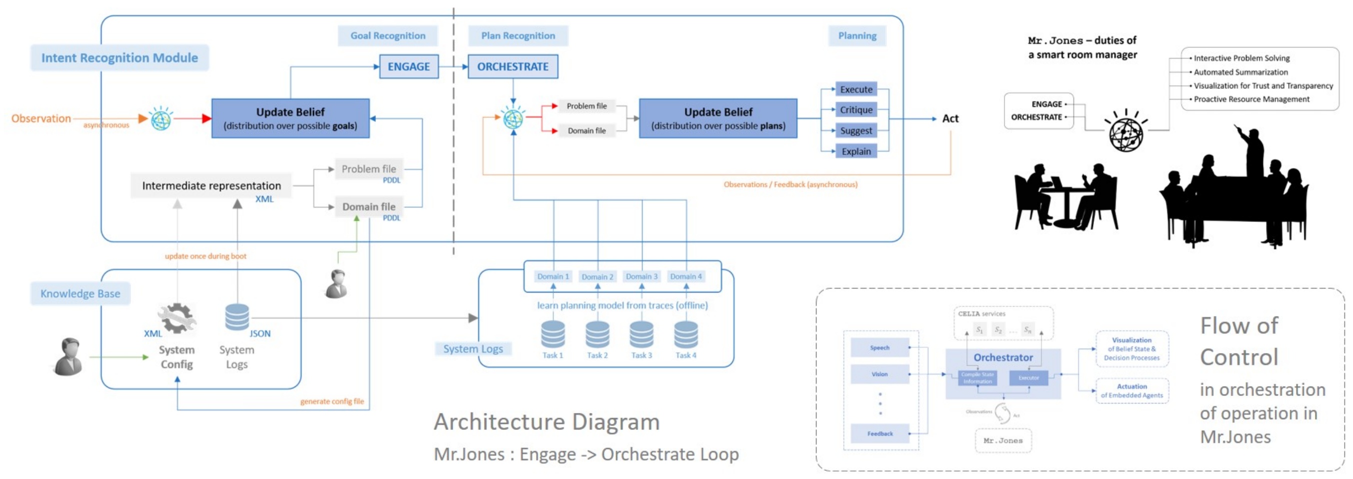

Intent Recognition Module
Goal Recognition
Plan Recognition
Planning
ENGAGE
ORCHESTRATE
Observation
asynchronous
Update Belief (distribution over possible goals)
Update Belief (distribution over possible plans)
Problem file
Domain file
Execute
Critique
Suggest
Explain
Act
Intermediate representation
XML
Problem file PDDL
Domain file PDDL
Observations / Feedback (asynchronous)
update once during boot
Knowledge Base
XML
JSON
System Config
System Logs
generate config file
System Logs
Domain 1
Domain 2
Domain 3
Domain 4
learn planning model from traces (offline)
Task 1
Task 2
Task 3
Task 4
Architecture Diagram
Mr.Jones : Engage -> Orchestrate Loop
Mr.Jones – duties of a smart room manager
ENGAGE
ORCHESTRATE
Interactive Problem Solving
Automated Summarization
Visualization for Trust and Transparency
Proactive Resource Management
CELIA services
Speech
Vision
Feedback
Orchestrator
Visualization of Belief State & Decision Processes
Actuation of Embedded Agents
Mr.Jones
Flow of Control
in orchestration of operation in Mr.Jones

Figure 1: Architecture diagram illustrating the building blocks of `Mr.Jones` – the two main components *Engage* and *Orchestrate* situates the agent proactively in a decision support setting with human decision makers in the loop. The top right inset shows the different roles of `Mr.Jones` as a smart room orchestrator and meeting facilitator. The bottom right inset illustrates the flow of control in `Mr.Jones` – each service runs in parallel and asynchronously to maintain anytime response of all the individual components.

`Mr.Jones`: An end-to-end planning system

We divide the responsibilities of `Mr.Jones` into two processes – ***Engage***, where plan recognition techniques are used to identify the task in progress; and ***Orchestrate***, which involves active participation in the decision-making process via real-time plan generation, visualization, and monitoring.

ENGAGE This consists of `Mr.Jones` monitoring various inputs from the world in order to situate itself in the context of the group interaction. First, the assistant gathers various inputs like speech transcripts, live images, and the positions of people within a meeting space; these inputs are fed into a higher level symbolic reasoning component. Using this, the assistant can (1) *requisition* resources and services that may be required to support the most likely tasks based on its recognition; (2) *visualize* the decision process – this can depict both the agent's own internal recognition algorithm, and an external, task-dependent process; and (3) *summarize* the group decision-making process.

ORCHESTRATE This process is the decision support assistant's contribution to the group's collaboration. This can be done using standard planning techniques, and can fall under the aegis of one of four actions as shown in Figure 1. These actions, some of which are discussed in more detail in (Sengupta et al. 2017), are: (1) *execute*, where the assistant performs an action or a series of actions related to the task at hand; (2) *critique*, where the assistant offers recommendations on the actions currently in the collaborative decision sequence; (3) *suggest*, where the assistant suggests new decisions and actions that can be discussed collaboratively; and (4) *explain*, where the assistant explains its rationale for adding or suggesting a particular decision. The *Orchestrate* process thus provides the "support" part of the decision support assistant. The *Engage* and *Orchestrate* processes can be seen as somewhat parallel to the *interpretation* and *steering* processes defined in the crowdsourcing scenarios of (Talamadupula et al. 2013; Manikonda et al. 2017). The difference in these new scenarios is that the humans are the final decision makers, with the assistant merely supporting the decision making.

Architecture Design & Key Components The central component – the Orchestrator[1] – regulates the flow of information and control flow across the modules that manage the various functionalities of the `CEL`; this is shown in Figure 1. These modules are mostly asynchronous in nature and may be: (1) services[2] processing sensory information from various input devices across different modalities like audio (microphone arrays), video (PTZ cameras / Kinect), motion sensors (Myo / Vive) and so on; (2) services handling the different services of `CEL`; and (3) services that attach to the `Mr.Jones` module. The Orchestrator is responsible for keeping track of the current state of the system as well as coordinating actuation either in the belief/knowledge space, or in the actual physical space.

Knowledge Acquisition / Learning The knowledge contained in the system comes from two sources – (1) the developers and/or users of the service; and (2) the system's own memory; as illustrated in Figure 1. One significant barrier towards the adoption of higher level reasoning capabilities into such systems has been the lack of familiarity of developers and end users with the inner working of these technologies. With this in mind we provide an XML-based modeling interface – i.e. a *"system config"* – where users can easily configure new environments. This information in turn enables automatic generation of the files that are internally required by the reasoning engines. Thus system specific information is bootstrapped into the service specifications written by expert

[1]Not to be confused with the term *Orchestrate* from the previous section, used to describe the phase of active participation.

[2]Built on top of the Watson Conversation and Visual Recognition services on IBM Cloud and other IBM internal services.

developers, and this composite knowledge can be seamlessly transferred across task domains and physical configurations. The granularity of the information encoded in the models depends on the task at hand – for example, during the *Engage* phase, the system uses much higher level information (e.g. identities of agents in the room, their locations, speech intents, etc.) than during the *Orchestrate* phase, where more detailed knowledge is needed. This enables the system to reason at different levels of abstraction independently, thus significantly improving the scalability as well as robustness of the recognition engine.

Plan Recognition The system employs the probabilistic goal / plan recognition algorithm from (Ramirez and Geffner 2010) to compute its beliefs over possible tasks. The algorithm casts the plan recognition problem as a planning problem by compiling away observations to the form of actions in a new planning problem. The solution to this new problem enforces the execution of these observation-actions in the observed order. This explainsmini the reasoning process behind the belief distribution in terms of the possible plans that the agent envisioned (as seen in Figure 2).

Plan Generation The FAST-DOWNWARD planner (Helmert 2006) provides a suite of solutions to the forward planning problem. The planner is also required internally by the Recognition Module when using the compilation from (Ramirez and Geffner 2010), or in general to drive some of the orchestration processes. The planner reuses the compilation from the Recognition Module to compute plans that preserve the current (observed) context.

Visualizations in Mr.Jones

The CEL is a smart environment, equipped with various sensors and actuators to facilitate group decision making. Automated planning techniques, as explained above, are the core component of the decision support capabilities in this setting. However, the ability to plan is rendered insufficient if the agent cannot communicate that information effectively to the humans in the loop. Dialog as a means of interfacing with the human decision makers often becomes clumsy due to the difficulty of representing information in natural language, and/or the time taken to communicate. Instead, we aim to build visual mediums of communication between the planner and the humans for the following key purposes –

- *Trust & Transparency* - Externalizing the various pathways involved in the decision support process is essential to establish trust between the humans and the machine, as well as to increase situational awareness of the agents. It allows the humans to be cognizant of the internal state of the assistant, and to infer decision rationale, thereby reducing their cognitive burden.
- *Summarization of Minutes* - The summarization process is a representation of the beliefs of the agent with regard to what is going on in its space over the course of an activity. Since the agent already needs to keep track of this information in order to make its decisions, we can replay or sample from it to generate an automated visual summary of (the agent's belief of) the proceedings in the room.

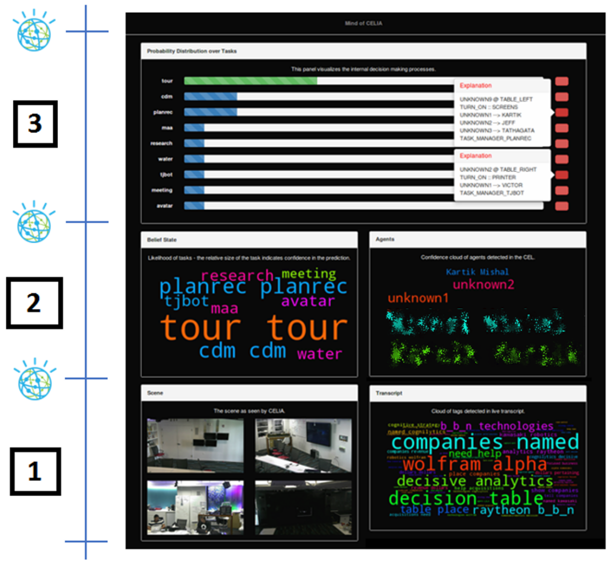


Figure 2: Snapshot of the mind of Mr.Jones externalizing different stages of its cognitive processes.

- *Decision Making Process* - Finally, and perhaps most importantly, the decision making process itself needs efficient interfacing with the humans – this can involve a range of things from showing alternative solutions to a task, to justifying the reasoning behind different suggestions. This is crucial in a mixed initiative planning setting (Horvitz 1999; 2007) to allow for human participation in the planning process, as well as for the planner's participation in the humans' decision making process.

Mind of Mr.Jones

First, we will describe the externalization of the "mind" of Mr.Jones – i.e. the various processes that feed the different capabilities of the agent. A snapshot of the interface is presented in Figure 2. The interface itself consists of five widgets. The largest widget on the top shows the various usecases that the CEL is currently set up to support. In the current CEL setup, there are nine such usecases. The widget represents the probability distribution that indicates the confidence of Mr.Jones in the respective task being the one currently being collaborated on, along with a button for the provenance of each such belief. The information used as provenance is generated directly from the plans used internally by the recognition module (Ramirez and Geffner 2010) and justifies why, given its model of the underlying planning problems, these tasks look likely in terms of plans that achieve those tasks. Model based algorithms are especially useful in providing explanations like this (Sohrabi, Baier, and McIlraith 2011; Fox, Long, and Magazzeni 2017). The system is adept at handling uncertainty in its inputs (it is interesting to note that in coming up with an explanatory plan it has announced likely assignments to unknown agents in its space). In Figure 2, Mr.Jones has placed the maximum confidence in the tour usecase.

Below the largest widget is a set of four widgets, each of which give users a peek into an internal component of Mr.Jones. The first widget, on the top left, presents a wordcloud representation of Mr.Jones's belief in each of the tasks; the size of the word representing that task corresponds to the probability associated with that task. The second widget, on the top right, shows the agents that are recognized as being in the environment currently – this information is used by the system to determine what kind of task is more likely. This information is obtained from four independent camera feeds that give Mr.Jones an omnispective view of the environment; this information is represented via snapshots (sampled at 10-20 Hz) in the third widget, on the bottom left. In the current example, Mr.Jones has recognized the agents named (anonymized) "XXX" and "YYY" in the scenario. Finally, the fourth widget, on the bottom right, represents a wordcloud based summarization of the audio transcript of the environment. This transcript provides a succinct representation of the things that have been said in the environment in the recent past via the audio channels. Note that this widget is merely a summarization of the full transcript, which is fed into the IBM Watson Conversation service to generate observations for the plan recognition module. The interface thus provides a real-time snapshot of the various sensory and cognitive organs associated with Mr.Jones- the eyes, ears, and mind of the CEL. The interface is organized at increasing levels of abstraction –

[1] *Raw Inputs* – These show the camera feeds and voice capture (speech to text outputs) as received by the system. These help in externalizing what information the system is working with at any point of time and can be used, for example, in debugging at the input level if the system makes a mistake or in determining whether it is receiving enough information to make the right decisions. It is especially useful for an agent like Mr.Jones, which is not embodied in a single robot or interface but is part of the environment as a whole. As a result of this, users may find it difficult to attribute specific events to the agent.

[2] *Lower level reasoning* – The next layer deals with the first stage of reasoning over these raw inputs – What are the topics being talked about? Who are the agents in the room? Where are they situated? This helps an user identify what knowledge is being extracted from the input layer and fed into the reasoning engines. It increases the situational awareness of agents by visually summarizing the contents of the scene at any point of time.

[3] *Higher level reasoning* – Finally, the top layer uses information extracted at the lower levels to reason about abstract tasks in the scene. It visualizes the outcome of the plan recognition process, along with the provenance of the information extracted from the lower levels (agents in the scene, their positions, speech intents, etc.). This layer puts into context the agent's current understanding of the processes in the scene.

Demonstration 1 We now demonstrate how the *Engage* process evolves as agents interact in the CEL. The demonstration begins with two humans discussing the CEL environment, followed by one agent describing a projection of the Mind of Mr.Jones on the screen. The other agent then discusses how a Mergers and Acquisitions (M&A) task (Kephart and Lenchner 2015) is carried out. *A video of this demonstration can be accessed at https://www.youtube.com/watch?v=ZEHxCKodEGs.* The video contains a window that demonstrates the evolution of the Mr.Jones interface through the duration of the interaction. This window illustrates how Mr.Jones's beliefs evolve dynamically in response to interactions in real-time.

Demonstration 2 After a particular interaction is complete Mr.Jones can automatically compile a summarization (or minutes) of the meeting by sampling from the visualization of its beliefs. *An anonymized video of a typical summary can be accessed at https://youtu.be/AvNRgsvuVOo.* This kind of *visual summary* provides a powerful alternative to established meeting summarization tools like text-based minutes. The visual summary can also be used to extract abstract insights about this one meeting, or a set of similar meetings together and allows for agents that may have missed the meeting to catch up on the proceedings. Whilst merely sampling the visualization at discrete time-intervals serves as a powerful tool towards automated summary generation, we anticipate the use of more sophisticated visualization (Dörk et al. 2010) and summarization (Shaw 2017; Kim, Chacha, and Shah 2015; Kim and Shah 2016) techniques in the future.

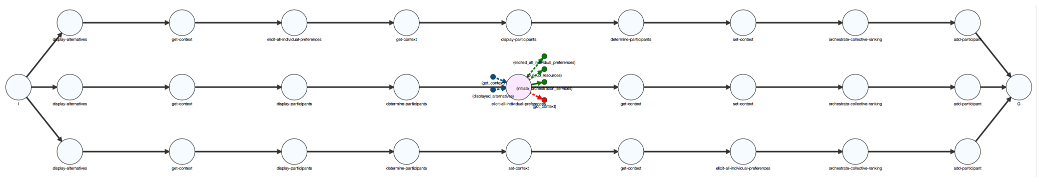
Model-Based Plan Visualization : Fresco

We start by describing the planning domain that is used in the rest of this section, followed by a description of Fresco's different capabilities in terms of *top-K plan visualization* and *model-based plan visualization*. We conclude by describing the implementation details on the back-end.

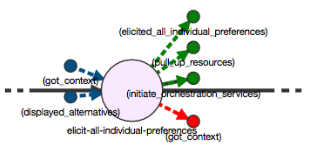
The Collective Decision Domain We use a variant of the Mergers and Acquisitions (M&A) task called *Collective Decision* (CD). The CD domain models the process of gathering input from a decision makers in a smart room, and the orchestration of comparing alternatives, eliciting preferences, and finally ranking of the possible options.

Top-K Visualization

Most of the automated planning technology and literature considers the problem of generating a single plan. Recently, however, the paradigm of Top-K planning (Riabov, Sohrabi, and Udrea 2014) has gained traction. Top-K plans are particularly useful in domains where producing and deliberating on multiple alternative plans that go from the same fixed initial state and the same fixed goal is important. Many decision support scenarios, including the one described above, are of this nature. Moreover, Top-K plans can also help in realizing unspecified user preferences, which may be very hard to model explicitly. By presenting the user(s) with multiple alternatives, an *implicit* preference elicitation can instead be performed. The Fresco interface supports visualization of the K top plans for a given problem instance and domain model, as shown in Figure 3a. In order to generate the Top-K plans, we use an experimental Top-K planner (Katz et al. 2018) that is built on top of Fast Downward (Helmert 2006).



(a) Top-K plan visualization showing alternative plans for a given problem.


(elicited_all_individual_preferences)
(pull_up_resources)
(got_context)
(initiate_orchestration_services)
(displayed_alternatives)
elicit-all-individual-preferences
(got_context)

(b) Action Descriptions

Figure 3: Visualization of plans in `Fresco` showing top-K alternative solutions (K=3) for a given planing problem (left) and on-demand visualization of each action in the plan (zoomed-in; right) in terms of causal links consumed and produced by it.

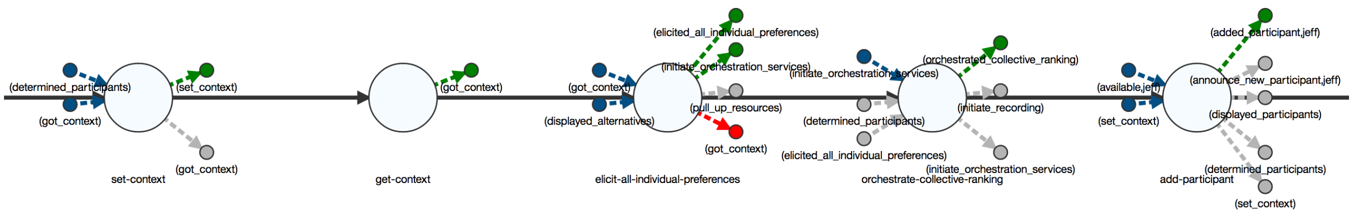

(determined_participants)
(got_context)
(set_context)
(got_context)
set-context
(got_context)
get-context
(elicited_all_individual_preferences)
(initiate_orchestration_services)
(got_context)
(displayed_alternatives)
(pull_up_resources)
(got_context)
elicit-all-individual-preferences
(initiate_orchestration_services)
(orchestrated_collective_ranking)
(determined_participants)
(initiate_recording)
(elicited_all_individual_preferences)
(initiate_orchestration_services)
orchestrate-collective-ranking
(added_participant,jeff)
(available,jeff)
(announce_new_participant,jeff)
(set_context)
(displayed_participants)
(determined_participants)
add-participant
(set_context)

Figure 4: Visualization as a process of explanation – minimized view of conditions relevant to a plan. Blue, green and red nodes indicate preconditions, add and delete effects respectively. The conditions which are not necessary causes for this plan (i.e. the plan is still optimal in a domain without these conditions) are grayed out in the visualization (11 out of a total 30).

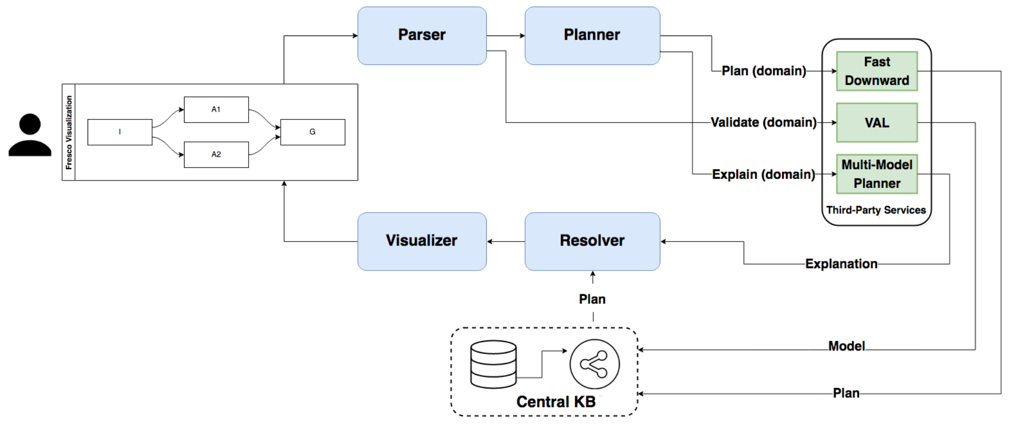
Model-based Plan Visualization

The requirements for visualization of plans can have different semantics depending on the task at hand – e.g. showing the search process that produced the plan, and the decisions taken (among possible alternative solutions) and tradeoffs made (by the underlying heuristics) in that process; or revealing the underlying domain or knowledge base that engendered the plan. The former involves visualizing the *how* of plan synthesis, while the latter focuses on the *why*, and is model-based and algorithm independent. Visualizing the how is useful to the developer of the system during debugging, but serves little purpose for the end user who would rather be told the rationale behind the plan: why is this plan better than others, what individual actions contribute to the plan, what information is getting consumed at each step, and so on. Unfortunately, much of the visualization work in the planning community has been confined to depicting the search process alone (Thayer 2010; 2012; Magnaguagno et al. 2017). `Fresco`, on the other hand, aims to focus on the *why* of a plan's genesis, in the interests of establishing common ground with human decision-makers. At first glance, this might seems like an easy problem – we could just show what the preconditions and effects are for each action along with the causal links in the plan. However, even for moderately sized domains, this turns into a clumsy and cluttered approach very soon, given the large number of conditions to be displayed. In the following, we will describe how `Fresco` handles this problem of overload.

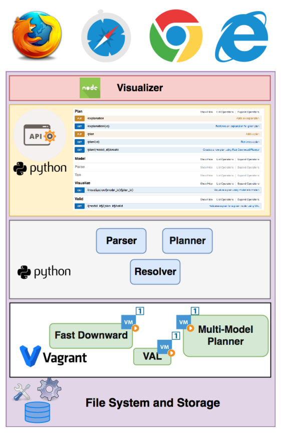
Visualization as a Process of Explanation We begin by noting that the process of visualization can in fact be seen as a *process of explanation*. In model-based visualization, as described above, the system is essentially trying to explain to the viewer the salient parts of its knowledge that contributed to this plan. In doing so, it is externalizing what each action is contributing to the plan, as well as outlining why this action is better that other possible alternatives.

Explanations in Multi-Model Planning Recent work has shown (Chakraborti et al. 2017a) how an agent can explain its plans to the user when there are differences in the models (of the same planning problem) of the planner and the user, which may render an optimal plan in the planner's model sub-optimal or even invalid–and hence unexplainable–in the user's mental model. An explanation in this setting constitutes a *model update* to the human such that the plan (that is optimal to the planner) in question also becomes optimal in the user's updated mental model. This is referred to as a *model reconciliation process* (MRP). The smallest explanation is called *minimally complete* (MCE).

Model-based Plan Visualization ≡ Model Reconciliation with Empty Model As we mentioned previously, exposing the entire model to the user is likely to lead to cognitive overload and lack of situational awareness due to the amount of information that is not relevant to the plan in question. We want to minimize the clutter in the visualization and yet maintain all relevant information pertaining to the plan. *We do this by launching an instantiation of the model reconciliation process with the planner's model and an empty model as inputs.* An empty model is a copy of the given model where actions do not have any conditions and the initial state is empty (the goal is still preserved). Following from the above discussion, the output of this process is then the minimal set of conditions in the original model that ensure optimality of the given plan. In the visualization, the rest of the conditions from the domain are grayed out. (Chakraborti et al. 2017a) showed how this can lead to a significant pruning of conditions that do not contribute to the generation of a particular



(a) Architecture diagram of `Fresco`.



(b) Software stack

Figure 5: Illustration of the flow of control (left) in `Fresco` between the plan generator (FD), explanation generator (MMP), and plan validator (VAL) with the visualization modules. The MMP code base is in the process of being fully integrated into `Fresco`, and it is currently run as a stand-alone component. The software stack (right) shows the infrastructure supporting `Fresco` in the backend.

plan. An instance of this process on the CD domain is illustrated in Figure 4.

Note that the above may not be the only way to minimize information being displayed. There might be different kinds of information that the user cares about, depending on their preferences. This is also highlighted by the fact that an MCE is not unique for a given problem. These preferences can be *learned* in the course of interactions.

Architecture of `Fresco` The architecture of `Fresco`, shown in Figure 5a, includes several core modules such as the parser, planner, resolver, and visualizer. These modules are all connected in a feed-forward fashion. The parser module is responsible for converting domain models and problem instances into python objects, and for validating them using VAL (Howey, Long, and Fox 2004). Those objects are then passed on to the planner module, which relies on Fast-Downward (FD) and the Multi-Model Planner (MMP) (Chakraborti et al. 2017a) to generate a plan along with its explanation. The resolver module consumes the plan, the explanation, and the domain information to not only ground the plan, but also to remove any preconditions, add, or delete effects that are deemed irrelevant by the MMP module. Finally, the visualizer module takes the plan from the resolver module as an input, and builds graphics that can be rendered within any well-known web browser. Our focus in designing the architecture was on making it functionally modular and configurable, as shown in Figure 5b. While the first three modules described above are implemented using Python, the visualizer module is implemented using Javascript and the `D3` graphics library. Our application stack uses `REST` protocols to communicate between the visualizer module and the rest of the architecture. We also accounted for scalability and reliability concerns by containerizing the application with `Kubernetes`, in addition to building individual containers / virtual machines for third party services like VAL, Fast-Downward, and MMP.

Work in Progress

While we presented the novel notion of *explanation as visualization* in the context of AI planning systems in this paper via the implemention of the `Mr.Jones` assistant, there is much work yet to be done to embed this as a central research topic in the community. We conclude the paper with a brief outline of future work as it relates to the visualization capabilities of `Mr.Jones` and other systems like it.

Visualization for Model Acquisition Model acquisition is one of the biggest impediments towards the adoption of planning technologies. Our own work with `Mr.Jones` is not immune to this problem. Although we have enabled an XML-based modeling interface, the next iteration of making this easily consumable for non-experts involves two steps: first, we impose an (possibly graphical) interface on top of the XML structure to obtain information in a structured manner. We can thenl provide visualizations such as those described in (Bryce et al. 2017) in order to help with iterative acquisition and refinement of the planning model.

Tooling Integration Eventually, our vision – not restricted to any one planning tool or technology – is to integrate the capabilities of `Fresco` into a domain-independent planning tool such as `planning.domains` (Muise 2016), which will enable the use of these visualization components across various application domains. `planning.domains` realizes the long-awaited planner-as-a-service paradigm for end users, but is yet to incorporate any visualization techniques for the user. Model-based visualization from `Fresco`, complemented with search visualizations from emerging techniques like `WebPlanner` (Magnaguagno et al. 2017), can be a powerful addition to the service.

Mixed-Reality Finally, recent advances in mixed-reality technologies provide exciting opportunities for newer modes of interaction with AI agents (Williams et al. 2018). We explore such capabilities in the space of decision-support in (Sengupta, Chakraborti, and Kambhampati 2018).

Acknowledgements A significant part of this work was initiated and completed while Tathagata Chakraborti was an intern at IBM's T. J. Watson Research Center during the summer of 2017. The continuation of his work at ASU is supported by an IBM Ph.D. Fellowship.

References

Benton, J.; Smith, D.; Kaneshige, J.; and Keely, L. 2017. CHAP-E: A plan execution assistant for pilots. In *Proceedings of the Workshop on User Interfaces and Scheduling and Planning*, UISP 2017, 1–7.

Bryce, D.; Bonasso, P.; Adil, K.; Bell, S.; and Kortenkamp, D. 2017. In-situ domain modeling with fact routes. In *Proceedings of the Workshop on User Interfaces and Scheduling and Planning*, UISP 2017, 15–22.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017a. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *IJCAI*.

Chakraborti, T.; Talamadupula, K.; Dholakia, M.; Srivastava, B.; Kephart, J. O.; and Bellamy, R. K. 2017b. Mr. Jones – Towards a Proactive Smart Room Orchestrator. *AAAI Fall Symposium on Human-Agent Groups*.

Dörk, M.; Gruen, D.; Williamson, C.; and Carpendale, S. 2010. A Visual Backchannel for Large-Scale Events. *IEEE Transactions on Visualization and Computer Graphics*.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable Planning. In *First IJCAI Workshop on Explainable AI (XAI)*.

Freedman, R. G., and Frank, J. D., eds. 2017. *Proceedings of the First Workshop on User Interfaces and Scheduling and Planning*. AAAI.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Horvitz, E. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 159–166. ACM.

Horvitz, E. J. 2007. Reflections on challenges and promises of mixed-initiative interaction. *AI Magazine* 28(2):3.

Howey, R.; Long, D.; and Fox, M. 2004. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, 294–301. IEEE.

Izygon, M.; Kortenkamp, D.; and Molin, A. 2008. A procedure integrated development environment for future spacecraft and habitats. In *Space Technology and Applications International Forum*.

Katz, M.; Sohrabi, S.; Udrea, O.; and Winterer, D. 2018. A Novel Iterative Approach to Top-k Planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Kephart, J. O., and Lenchner, J. 2015. A symbiotic cognitive computing perspective on autonomic computing. In *Autonomic Computing (ICAC), 2015 IEEE International Conference on*, 109–114.

Kim, J., and Shah, J. A. 2016. Improving team's consistency of understanding in meetings. *IEEE Transactions on Human-Machine Systems* 46(5):625–637.

Kim, B.; Chacha, C. M.; and Shah, J. A. 2015. Inferring team task plans from human meetings: A generative modeling approach with logic-based prior. *Journal of Artificial Intelligence Research*.

Magnaguagno, M. C.; Pereira, R. F.; Móre, M. D.; and Meneguzzi, F. 2017. Web planner: A tool to develop classical planning domains and visualize heuristic state-space search. *ICAPS 2017 User Interfaces for Scheduling & Planning (UISP) Workshop*.

Manikonda, L.; Chakraborti, T.; Talamadupula, K.; and Kambhampati, S. 2017. Herding the crowd: Using automated planning for better crowdsourced planning. *Journal of Human Computation*.

Muise, C. 2016. Planning.Domains. In *The 26th International Conference on Automated Planning and Scheduling - Demonstrations*.

Myers, K. L. 1996. Advisable planning systems. *Advanced Planning Technology* 206–209.

Ramirez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *AAAI*.

Riabov, A.; Sohrabi, S.; and Udrea, O. 2014. New algorithms for the top-k planning problem. In *Proceedings of the Scheduling and Planning Applications woRKshop (SPARK) at the 24th International Conference on Automated Planning and Scheduling (ICAPS)*, 10–16.

Sengupta, S.; Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2017. RADAR - A Proactive Decision Support System for Human-in-the-Loop Planning. In *AAAI Fall Symposium on Human-Agent Groups*.

Sengupta, S.; Chakraborti, T.; and Kambhampati, S. 2018. MA-RADAR – A Mixed-Reality Interface for Collaborative Decision Making. *ICAPS UISP*.

Shaw, D. 2017. How Wimbledon is using IBM Watson AI to power highlights, analytics and enriched fan experiences. `https://goo.gl/r6z3uL`.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred explanations: Theory and generation via planning. In *AAAI*.

Talamadupula, K.; Kambhampati, S.; Hu, Y.; Nguyen, T.; and Zhuo, H. H. 2013. Herding the crowd: Automated planning for crowdsourced planning. In *HCOMP*.

Thayer, J. 2010. Search Visualizations. `https://www.youtube.com/user/TheSuboptimalGuy`.

Thayer, J. T. 2012. Heuristic search under time and quality bounds. *Ph. D. Dissertation, University of New Hampshire*.

Williams, T.; Szafir, D.; Chakraborti, T.; and Ben Amor, H. 2018. Virtual, augmented, and mixed reality for human-robot interaction. In *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 403–404. ACM.

Projection-Aware Task Planning and Execution for Human-in-the-Loop Operation of Robots in a Mixed-Reality Workspace [*] [†]

Tathagata Chakraborti and **Sarath Sreedharan** and **Anagha Kulkarni** and **Subbarao Kambhampati**
Arizona State University, Tempe, AZ 85281 USA
{ tchakra2, ssreedh3, akulka, rao } @ asu.edu

Abstract

Recent advances in mixed-reality technologies have renewed interest in alternative modes of communication for human-robot interaction. However, most of the work in this direction has been confined to tasks such as teleoperation, simulation or explication of individual actions of a robot. In this paper, we will discuss how the capability to project intentions affect the *task planning* capabilities of a robot. Specifically, we will start with a discussion on how projection actions can be used to reveal information regarding the future intentions of the robot at the time of task execution. We will then pose a new planning paradigm – *projection-aware planning* – whereby a robot can trade off its plan cost with its ability to reveal its intentions using its projection actions. Finally, we will show how in the context of task planning, projection actions may also be useful for plan explicability and explanations. We will demonstrate each of these scenarios with the help of a joint human-robot activity using the HoloLens.

1 Introduction

Effective planning for human robot teams not only requires the capacity to be "human-aware" during the plan generation process, but also the ability to interact with the human during the plan execution phase. Prior work has underlined this need (Karpas et al. 2015) as well as explored ways to exchange (Tellex et al. 2014; Chakraborti et al. 2017c) information in natural language during interaction with the human in the loop. However, the state of the art in natural language considerably limits the scope of such interactions, especially where precise instructions are required. In this paper, we present the case of wearable technologies (e.g. HoloLens) for effective *communication of intentions* during human-in-the-loop operation of robots.

The last decade has seen a massive increase in robots deployed on the factory floor (Robotenomics 2017). This has led to fears of massive job loss for humans in the manufacturing industry, as well concerns of security of jobs that do remain. The latter is not an emerging concern, though. Automation of the manufacturing industry has gone hand in hand with incidents of misaligned intentions between the robots and their human co-workers, leading to at least four instances of fatality (Weiss 2015). This dates back to as early as 1979 when a robot arm crushed a worker to death while gathering supplies in the Michigan Ford Motor Factory, to as recent as 2015 in a much publicized accident at the Volkswagen factory in Baunatal, Germany. With 1.3 million new robots predicted to enter the workspace by next year (PRNewswire 2016), such concerns are only expected to escalate. A closer look at the dynamics of employment in the manufacturing industry, however, reveals that the introduction of automation has in fact increased productivity (Muro and Andes 2015) as well as, surprisingly, contributed to a steady increase in the number of jobs for human workers (Look 2016) in Germany (which dominates in terms of deployed robots in the industry). We posit then either a semi-autonomous workspace in future with increased hazards due to misaligned interests of robots in the shared environment, or a future where the interests of the human workers will be compromised in favor of automation. In light of this, it

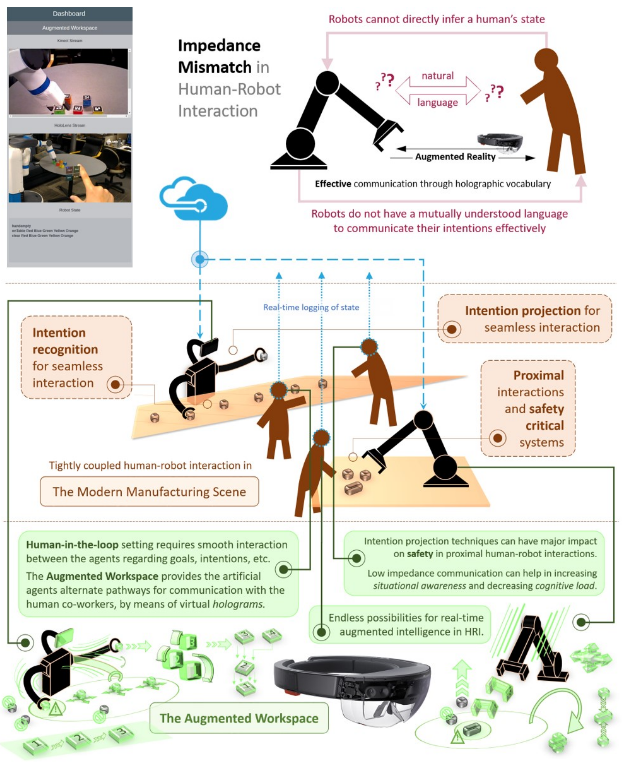

Dashboard
Augmented Workspace
Kinect Stream
HoloLens Stream
Robot State
Impedance Mismatch in Human-Robot Interaction
Robots cannot directly infer a human's state
natural language
Augmented Reality
Effective communication through holographic vocabulary
Robots do not have a mutually understood language to communicate their intentions effectively
Real-time logging of state
Intention recognition for seamless interaction
Intention projection for seamless interaction
Proximal interactions and safety critical systems
Tightly coupled human-robot interaction in
The Modern Manufacturing Scene
Human-in-the-loop setting requires smooth interaction between the agents regarding goals, intentions, etc.
The Augmented Workspace provides the artificial agents alternate pathways for communication with the human co-workers, by means of virtual holograms.
Intention projection techniques can have major impact on safety in proximal human-robot interactions.
Low impedance communication can help in increasing situational awareness and decreasing cognitive load.
Endless possibilities for real-time augmented intelligence in HRI.
The Augmented Workspace

Figure 1: A cloud-based distributed augmented workspace to combat impedance mismatch in human robot interactions.

[*] Parts of this project appeared in the U.S. Finals of the Microsoft Imagine Cup 2017. Details: http://www.ae-robots.com/

[†] An update version is available at https://goo.gl/ZFGjPK

is essential that the next-generation factory floor (Automation World 2016) is able to adapt to these new technologies. Indeed, recent reports (Linette Lopez 2018) have hinted at significant gains to be had from bridging this gap.

At the core of this problem is the impedance mismatch between humans and robots in how they represent and communicate information. Despite the progress made in natural language processing, natural language understanding is still a largely unsolved problem, and as such robots find it difficult to express their own goals and intentions effectively. Thus there exists a significant communication barrier to be overcome from either side. While this may not always be a serious concern for deploying completely autonomous agents in isolated environments such as for space or underwater exploration, the priorities change considerably when humans and robots are involved in collaborative tasks, especially for concerns of safety, if not just to improve the effectiveness of collaboration. This is emphasized in the *Roadmap for U.S. Robotics* (Christensen et al. 2009) – *"humans must be able to read and recognize robot activities in order to interpret the robot's understanding"*.

Related Work The concept of intention projection for autonomous systems has, of course, been explored before. An early attempt was made in (Sato and Sakane 2000) in a prototype Interactive Hand Pointer (IHP) to control a robot in the human's workspace. Similar systems have since been developed to visualize trajectories of mobile wheelchairs and robots (Watanabe et al. 2015; Chadalavada et al. 2015), which suggest that humans prefer to interact with a robot when it presents its intentions directly as visual cues. The last few years have seen active research (Omidshafiei et al. 2015; 2016; Shen, Jin, and Gans 2013; Ishii et al. 2009; Mistry et al. 2010; Leutert, Herrmann, and Schilling 2013; Turk and Fragoso 2015; Maurtua et al. 2016) in this area, but most of these systems were passive, non-interactive and quite limited in their scope, and did not consider the state of the objects or the context of the plan pertaining to the action while projecting information. As such, the scope of intention projection has remained largely limited. Indeed, recent works (Andersen et al. 2016; Chakraborti et al. 2017a) have made the first steps towards extending these capabilities to the context of task planning and execution, but fall short of formalizing the notion of intention projections beyond the current action under execution.

Instead, in this paper, we demonstrate a system that is able to provide much richer information to the human during collaboration, in terms of the current state information, action being performed as well as future parts of the plan under execution, particularly with the notion of explicating or foreshadowing future intentions. Recent advances (Williams et al. 2018) in the field of mixed reality make this form of online interactive plan explication particularly compelling.

Note that the ability to communicate information, and planning with the knowledge of that ability when it is useful to disambiguate intentions, is not necessarily unique to mixed-reality interactions only. One could use the planner introduced in Section 4.4 to generate *content* for traditional speech-based interactions as well (c.f. recent works on verbalization of intentions in natural language (Tellex et al. 2014)). However, as demonstrated in this paper, the medium of mixed-reality provides a particularly concise and effective alternative vocabulary of communication, especially in more structured scenarios such as in collaborative manufacturing.

Recent work in the scope of human-aware task and motion planning has focused on generation of legible motion plans (Dragan and Srinivasa 2013) and explicable task plans (Zhang et al. 2017; Kulkarni et al. 2016) with the notion of trading off cost of plans with how easy they are to interpret for a human observer. This runs parallel to our work on intention projections. Note that, in effect, either during the generation or the execution of a plan, we are, in fact, trying to optimize the same criterion. However, in our case, the problem becomes much more intriguing since the robot gets to *enforce legibility or explicability of a plan by foreshadowing of actions that have not been executed yet*. Indeed, this connection has also been hinted at in recent work (Gong and Zhang 2018). *However, to the best of our knowledge, this is the first task-level planner to achieve this trade-off.*

The plan explanations and explicability process forms a delicate balancing act, as we have investigated in recent work (Chakraborti, Sreedharan, and Kambhampati 2018). This also has interesting implications to the intention projection ability as we demonstrate in the final section.

Similarly, in recent work (MacNally et al. 2018), authors have looked at the related problem of *"transparent planning"* where a robot tries to signal its intentions to an observer by performing disambiguating actions in its plan. Intention projection in the medium of mixed-reality is likely to be a perfect candidate for this purpose without incurring unnecessary cost of execution.

Contributions Thus the contributions of our paper are –

- In Sections 4 and 5, we demonstrate how an *Augmented Workspace* can assist in *task-level* planning and execution in collaborative human-robot interactions.
- In Section 4, we show how the intention projection techniques can be used to reduce **ambiguity over possible plans** during execution.
  - In Section 4.4, we show how this can be used to realize a *first of its kind* task planner that instead of considering only cost optimal plans, *generates* plans which are easier to explicate using intention projection actions.
- In Section 5, we demonstrate how the ability to project world information applies to the process of explanations for **inexplicability of a plan** during execution.

2 The Augmented Workspace

Our primary focus here is on structured settings like the manufacturing environment where wearables can be a viable solution for improving the workspace. Indeed, a reboot of the safety helmet and goggles only requires retro-fitting existing wearables with sensors that can enable these new technologies. Imagine, then, a human and robot engaged in an assembly task, where they are constructing a structure collaboratively. Further suppose that the human now needs a tool from the shared workspace. At this time, neither agent is sure what tools and objects the other is going to access in the immediate future - this calls for seamless transfer of relevant information without loss of workflow. Existing (general purpose) solutions will suggest intention recognition or natural language communication as a means to respond to this situation. While natural language and intent or gesture recognition techniques remain the ideal, and sometimes the only, choice (such as assistive robots that need to interact in daily settings), we note that these are inherently noisy and ambiguous and need not necessarily be the medium of choice in controlled environments such as on the factory floor or by the assembly line where the workspace can be engineered to enforce protocols in the interests of safety and productivity, in the form of safety helmets integrated with wearable technology (Ruffaldi et al. 2016).

Instead, in our proposed system, the robot projects its intentions as *holograms* thus making them directly accessible to the human in the loop, e.g. by projecting a pickup symbol on a tool it might use in future. Further, unlike in traditional mixed reality projection systems, the human can directly interact with these holograms to make his own intentions known to the robot, e.g. by gazing at and selecting the desired tool thus forcing the robot to replan. To this end, we develop, with the power of the HoloLens, an alternative communication paradigm that is based on the projection of explicit visual cues pertaining to the plan under execution via holograms such that they can be intuitively understood and directly read by the human partner. The "real" shared human-robot workspace is now thus augmented with the virtual space where the physical environment is used as a medium to convey information about the intended actions of the robot, the safety of the work space, or task-related instructions. We call this the *Augmented Workspace*. Recent development of augmented reality techniques (Williams et al. 2018) has opened up endless possibilities in such modes of communication. In this paper, we will use the classic (International Planning Competition 2011) BlocksWorld domain as a proxy to a collaborative assembly domain. Here the robot is tasked with making words (or configurations) out of lettered (or colored) blocks using stacking and unstacking actions, to mimic assembly of specified structures. This will be used as the domain to illustrate various use cases and demonstrations in the rest of the paper.

3 Preliminaries of AI Planning

A Classical Planning Problem (Chakraborti et al. 2017b) is a tuple $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$ with domain $\mathcal{D} = \langle F, A \rangle$ - where F is a set of fluents that define a state $s \subseteq F$, and A is a set of actions - and initial and goal states $\mathcal{I}, \mathcal{G} \subseteq F$. Action $a \in A$ is a tuple $\langle c_a, \mathit{pre}(a), \mathit{eff}^{\pm}(a) \rangle$ where c_a is the cost, and $\mathit{pre}(a), \mathit{eff}^{\pm}(a) \subseteq F$ are the preconditions and add/delete effects, i.e. $\delta_{\mathcal{M}}(s, a) \models \bot$ *if* $s \not\models \mathit{pre}(a)$; *else* $\delta_{\mathcal{M}}(s, a) \models s \setminus \mathit{eff}^{-}(a) \cup \mathit{eff}^{+}(a)$ where $\delta_{\mathcal{M}}(\cdot)$ is the transition function. The cumulative transition function is $\delta_{\mathcal{M}}(s, \langle a_1, a_2, \ldots, a_n \rangle) = \delta_{\mathcal{M}}(\delta_{\mathcal{M}}(s, a_1), \langle a_2, \ldots, a_n \rangle)$.

Note that the "model" $\mathcal{M}$ of a planning problem includes the action model *as well as the initial and goal states of an agent*. The solution to $\mathcal{M}$ is a sequence of actions or a (satisficing) *plan* $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ such that $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$. The cost of a plan π is $C(\pi, \mathcal{M}) = \sum_{a \in \pi} c_a$ if $\delta_{\mathcal{M}}(\mathcal{I}, \pi) \models \mathcal{G}$; ∞ otherwise. The cheapest plan $\pi^* = \arg\min_{\pi} C(\pi, \mathcal{M})$ is the (cost) optimal plan with cost $C^*_{\mathcal{M}}$.

Projection actions can include information regarding either the state of the world or the robot's plans – both of these can reveal information regarding the robot's future intentions, i.e. goals or plans. In this work, we assume a very simple projection model based on the truth value of specified conditions in parts of the plan yet to be executed –

An Action Projection AP is defined as a mapping $u : [0 \ldots |\pi|] \times A \mapsto \{\mathtt{T}, \mathtt{F}\}$ indicating $\exists\, j \geq i$ where $a_i, a_j \in \pi$ if $u(j, a_i) = \mathtt{T}$ and $a_i, a_j \not\in \pi$ otherwise – i.e. existence or membership of an action in the rest of the plan.

A State Value Projection SVP is defined as a mapping $v : F \times A \mapsto \{\mathtt{T}, \mathtt{F}\}$ so that there exists a state in the state sequence induced by the sub-plan starting from a_i where the state variable $f \in F$ holds the value $v(f, a_i)$, i.e. $\exists\, s' : \delta_{\mathcal{M}}(s, \pi') \models s'$ where s is the current state and π' is the sub-plan $(\pi)_{k=i}^{k \leq |\pi|}$ and $f \in s'$ iff $v(f, a_i) = \mathtt{T}$, $f \not\in s$ otherwise.

In the following sections, we will discuss how a robot can determine *when* to deploy *which* of these projections in order to better explicate its plans to a human in the loop.

4 Projections for Ambiguous Intentions

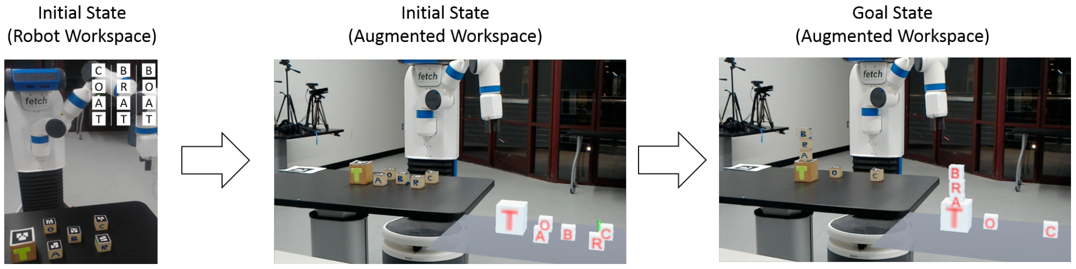
In this section, we will concentrate upon how projection actions can resolve ambiguity with regards to the intentions of a robot in the course of execution of a task plan.

4.1 Projection-Aware Plan Execution

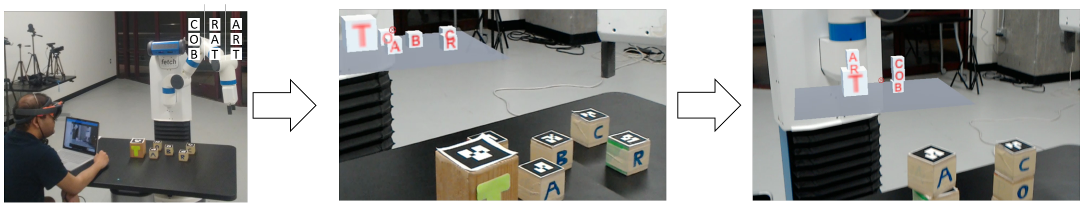
The first topic of consideration is the projection of intentions of a robot with a human *observer* in the loop.

Illustrative Example. Consider a robot involved in a mock assembly task (i.e. block stacking) as shown in Figure 2a. We will be using this setting throughout the rest of the paper. Here, the robot's internal goal is to form the word `BRAT`. However, given the letters available to it, it can form other words as well – consider two more possible goals `BOAT` and `COAT`. As such, it is difficult to say, from the point of view of the observer, by looking at the starting configuration, which of these is the real outcome of the impending plan. The robot can, however, at the start of its execution, choose to indicate that it has planned to pick up the block R later (by projecting a bobbing arrow on top of it), thereby resolving this ambiguity. Note that directly displaying the actual goal – here, the final word – is not possible in general across different domains. A video demonstrating this can be viewed at `https://goo.gl/SLgCPE`.

A Projection-Aware Plan Execution Problem PAPEP is a tuple $\Phi = \langle \pi, \Pi, \{AP\}, \{SVP\} \rangle$ where π is the robot's plan up for execution, Π (including π) is the set of possible plans it can execute, and $\{AP\}$ and $\{SVP\}$ are the set of action and state value projections available (i.e. due to π).



(a) The robot projects (AP) a green arrow to indicate pickup of a block that is part of an optimal plan to only one of its possible goals.



(b) The robot inverts the projection context and this time shows (SVP) which block is *not* going to be available using a red cross.

Figure 2: Projection-Aware Plan Execution for human observer and human-in-the-loop scenarios.

The solution to Φ is a composite plan $\pi^c \circ \pi$ where $\pi^c \subseteq \{AP\} \cup \{SVP\}$ are the projection actions that disambiguate the plans at the time of execution. We compute this using the concept of resource profiles, as introduced in (Chakraborti et al. 2016). Informally, a **resource** (Chakraborti et al. 2016) is defined as any state variable whose binary value we want to track. We will use this concept to tie each action or state value projection action to a single resource variable, whose effect can be monitored. For example, a `not-clear` predicate will indicate that a block is in use or not available while an action that produces or negates that predicate – e.g. `pick-up` can be similarly tracked through it. This mapping between projection actions and the corresponding resource variables is domain-dependent knowledge that is provided.

A Resource Profile $\mathcal{R}^\pi$ induced by a plan π on a resource r is a mapping $\mathcal{R}^\pi : [0 \ldots |\pi|] \times r \mapsto \{0, 1\}$, so that r is *locked* by π at step i if $\mathcal{R}^\pi(r, i) = 1$ and it is free otherwise.

A Cumulative Resource Profile $\mathbb{R}^\Pi$ induced by a *set* of plans Π on a resource r is a mapping $\mathbb{R}^\Pi : [0 \ldots \max_{\pi \in \Pi} |\pi|] \times r \mapsto [0, 1]$, so that r is *locked* with a probability $\mathbb{R}^\Pi(r, i) = \sum_{\pi \in \Pi} \mathcal{R}^\pi(r, i) \times P(\pi)$, where $P(\pi)$ is the prior probability of plan π (assumed uniform).

The set of projection actions π^c in the solution π^* to the PAPEP Φ are found by computing –

$$\arg\min_r \sum_{\pi \in \Pi} \mathcal{R}^{\pi^*} \times \mathbb{R}^\Pi \tag{1}$$

Thus, we are post-processing to minimize the conflicts between the current plan and other possible plans, so that the projection actions that are tied to the resources with the minimal conflicts give us the most distinguishing projection.

4.2 Projection-Aware Human-in-the-Loop Execution

In the previous example, we confined ourselves to situations with the human only as the observer. Now, we consider a situation where both the human and the robot are involved in task planning in a collaborative sense, i.e. both the human and the robot perform actions in a joint plan to achieve their goals which may or may not be shared.

Illustrative Example. Going back to the running example of the block stacking task, now consider that the robot and the human both have goals to make a three letter word out of `ART`, `RAT` and `COB` (as seen in Figure 2b). The robot has decided to make the word `ART`, but realizes that this leaves the human undecided on how to proceed. Thus the disambiguating projection action here includes annotating the `R` block with a "not available" symbol so that the only possible goal left for the human is `COB`. A video demonstrating this can be viewed at `https://goo.gl/SLgCPE` (same as in Section 4.1). Note that in this case the robot, in coming up with a useful projection action, has reversed the perspective from what is relevant to its own plan, to information that negates possible plans of the human in the loop.

A Projection-Aware Human-in-the-Loop Plan Execution Problem PAHILPEP is defined as the tuple $\Psi = \langle \Pi^R, \Pi^H, \mathbb{G}, \{AP\}, \{SVP\} \rangle$ where Π^R and Π^H are the set of possible plans the robot and the human can execute respectively, $\mathbb{G}$ is their shared team goal, and $\{AP\}$ and $\{SVP\}$ are the set of action and state value projections available to the robot (i.e. induced by Π^R).

The solution to Φ is, as before, a composite plan $\pi^c \circ \pi^R$ where the projection actions are composed with the robot's component of the joint team plan, such that $\delta(\mathcal{I},\ \pi^c \circ \pi^R \circ \pi^H) \models \mathbb{G}$. The set of projection actions π^c in the solution to the PAHILPEP Ψ is again found by computing –

$$\arg\max_r \mathbb{R}^{\Pi^H} \times \mathbb{R}^{\Pi^R} \tag{2}$$

Notice the inversion to `argmax`, since in the case of an active human in the loop, so as to provide the most pertinent information regarding conflicting intentions to the human.

Remark. Joint plans (Chakraborti et al. 2015) to reason over

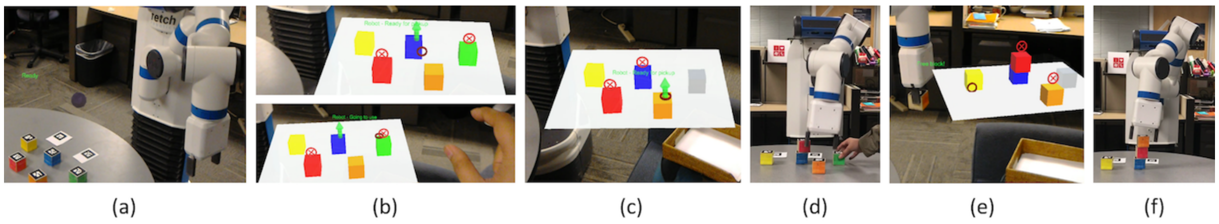


Figure 3: Interactive execution of a plan in the Augmented Workspace - **(a)** the robot wants to build a tower of height three with blocks blue, red and green. **(b)** Blocks are annotated with intuitive holograms, e.g. an upward arrow on the block the robot is going to pick up immediately and a red cross mark on the ones it is planning to use later. The human can also gaze on an object for more information (in the rendered text). **(c)** & **(d)** The human pinches on the green block and claims it for himself. The robot now projects a faded out green block and re-plans online to use the orange block instead (as evident by pickup arrow that has shifted on the latter at this time). **(e)** Real-time update and rendering of the current state showing status of the plan and objects in the environment. **(f)** The robot completes its new plan using the orange block.

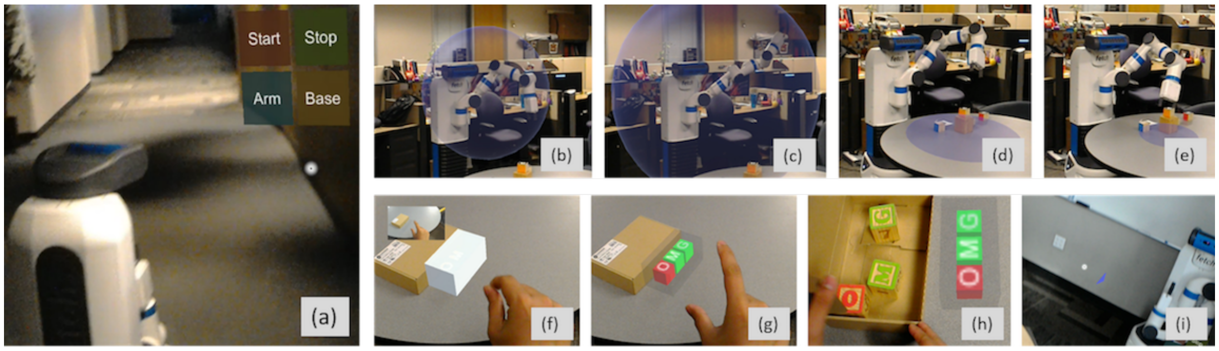


Figure 4: Interactive plan execution using the **(a)** Holographic Control Panel. Safety cues showing dynamic real-time rendering of volume of influence **(b) - (c)** or area of influence **(d) - (e)**, as well as **(i)** indicators for peripheral awareness. Interactive rendering of hidden objects **(f) - (h)** to improve observability and situational awareness in complex workspaces.

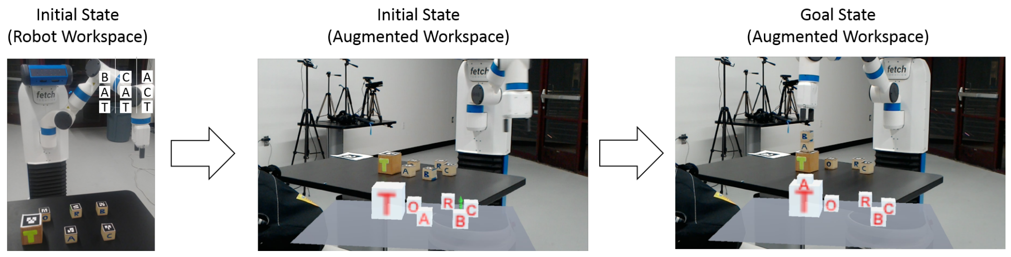
different modes of human-robot interactions has been investigated before, particularly in the context of using resource profiles (Chakraborti et al. 2016) for finding conflicts in the human's and the robot's plans. It is interesting to note the reversed dynamics of interaction in the example provided above – i.e. in (Chakraborti et al. 2016) the resource profiles were used so that the robot could replan based on probable conflicts so as to preserve the expected plans of the human. Here, we are using them to identify information to project to the human, so that the latter can replan instead.

4.3 Closing the Loop – Interactive Plan Execution

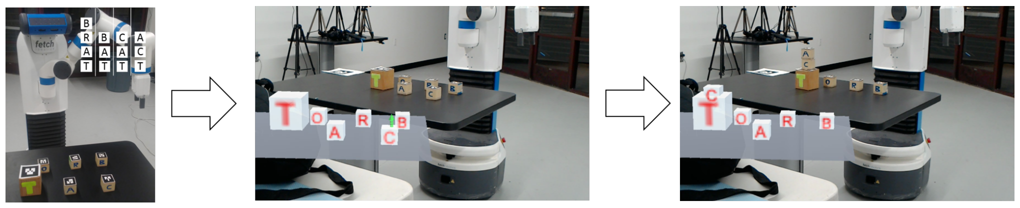
Of course, it may not be possible to always disentangle plans completely towards achievement of a shared goal in a collaborative setting. In the next demonstration, we show how the communication loop is closed by allowing the humans to interact directly with the holograms in the augmented workspace and spawn replanning commands to be handled by the robot, in the event of conflicting intentions.

Replanning – In the previous examples, the robot projected annotations onto the objects it is intending to manipulate into the human's point of view with helpful annotations or holograms that correspond to its intentions to use that object. The human can, in turn, access or claim a particular object in the virtual space and force the robot to re-plan, without there ever being any conflict of intentions in the real space. The humans in the loop can thus not only infer the robot's intent immediately from these holographic projections, but can also interact with them to communicate their own intentions directly and thereby modify the robot's behavior online. The robot can also then ask for help from the human, using these holograms. Figure 3 demonstrates one such scenario. The human can also go into finer control of the robot by accessing the Holographic Control Panel, as seen in Figure 4(a). The panel provides the human controls to start/stop execution of the robot's plan, as well as achieve fine grained motion control of both the base and the arm by making it mimic the user's arm motion gestures on the MoveArm and MoveBase holograms attached to the robot.

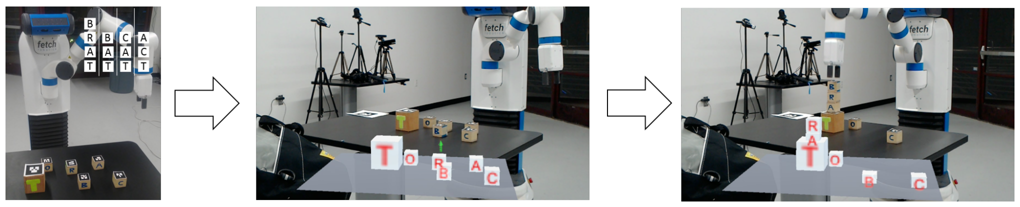
Assistive Cues – The use of AR is, of course, not just restricted to procedural execution of plans. It can also be used to annotate the collaborative workspace with artifacts derived from the current plan under execution in order to improve the fluency of collaboration. For example, Figure 4(b-e) shows the robot projecting its area of influence in its workspace either as a 3D sphere around it, or as a 2D circle on the area it is going to interact with. This is rendered dynamically in real-time based on the distance of the end effector to its center, and to the object to be manipulated. This can be very useful in determining safety zones around a robot in operation. As seen in Figure 4(f-i), the robot can also render



(a) Generating a plan that has the most discriminating projection (green arrow on `B` – only one word `BAT` possible).



(b) Trading-off cost of plan with ambiguity ($\alpha = 100$; green arrow on `C` – two words `ACT` and `CAT` possible).



(c) Trading-off cost of plan with ambiguity ($\alpha = 1000$; green arrow on `R` – only one (longer) word `BRAT` possible).

Figure 5: Projection-aware plan generation illustrating trade-off in plan cost and goal ambiguity during execution.

hidden objects or partially observable state variables relevant to a plan, as well as indicators to improve peripheral vision of the human, to improve their situational awareness.

Demonstrations for Sections 4.3 and 4.3 can be viewed at `https://goo.gl/pWWzJb`.

4.4 Projection-Aware Plan Generation

Now that we have demonstrated how intention projection can be used to disambiguate possible tasks at the time of execution, we ask *is it possible to use this ability to **generate** plans that are easier to disambiguate in the first place?*

Illustrative Example. Consider again the blocks stacking domain, where the robot is yet to decide on a plan, but it has three possible goals `BAT`, `CAT` and `ACT` (as seen in Figure 5a). From the point of view of cost optimal planning, all these are equally good options. However, notice that the letter `B` is in only one of the words, while the others are in at least two possible words. Thus the robot is able to *reduce the ambiguity in plans* by choosing the word `BAT` as a means of achieving the goal of making a word from the given set.

Illustrative Example. Now imagine that we have extended the possible set of words { `BAT`, `CAT`, `ACT` } with a longer word `BRAT`. The robot responds by projecting `R` and completes this longer word now, given `R` is the most discriminating action, and the possibility of projecting it ahead completely reveals its intentions **even though it involves the robot doing a longer and hence costlier plan** as seen in Figure 5c. This trade-off in the cost of plans and the ambiguity of intentions forms the essence of what we refer to as *projection-aware planning*. In fact, we can show that by correctly calibrating this trade-off, we can achieve different sweet spots in how much the robot decides to foreshadow disambiguating actions. As seen in Figure 5b, in cases where the action costs are relatively greater than gains due to resolved ambiguity, the robot achieves a middle-ground of generating a plan that has the same cost as the optimal plan to achieve the goal of making a word from this set, but also involves reasonable forecasting of (two) possible goals by indicating a future `pick-up` action on `C`. A video demonstration can be viewed at `https://goo.gl/bebtWS`.

A Projection-Aware Planning Problem PAPP is defined as the tuple $\Lambda = \langle \mathcal{M}, \kappa, \{AP\}, \{SVP\} \rangle$ where $\mathcal{M}$ is a planning problem and κ is a set of disjunctive landmarks.

The solution to Λ is a plan such that –

- π achieves the goal; and
- commitments imposed by the projection actions, i.e. future state conditions indicated by SVPs or actions promised by APs (Section 3) are respected.

The search for which projection actions to include is achieved by modifying a standard A-star search[1] (Hart, Nilsson, and Raphael 1968; Chakraborti et al. 2017b) so that the

[1]Note that to speed up search we used "outer entanglement" analysis (Chrpa and Barták 2009) to prune unnecessary actions for the blocks stacking domain.

objective function trades off actions costs and ambiguity of future plans (to possible landmarks) from the current state (as shown in Algorithm 1). This is given by –

$$\alpha\, C(\hat{\pi}, \mathcal{M}) + \beta\, \mathbb{E}(\Pi, \hat{\pi}) \tag{3}$$

Here Π is a set of possible plans that the robot can pursue from the current state. These are the top-K plans (Riabov, Sohrabi, and Udrea 2014) to the landmarks (Karpas and Domshlak 2009) in the domain[2]. $\mathbb{E}(\Pi)$ is the entropy of the probability distribution (Ramırez and Geffner 2010) over the plan set Π given the current plan prefix $\hat{\pi}$ to that state. Since a full evaluation of the plan recognition problem in every node is prohibitively expensive, we use a simple observation model where the currently proposed projection action tests *membership* of its parent action if it is an AP (or state value if it is an SVP) in the delete-relaxed plans (Bryce and Kambhampati) to each landmark –

$$\alpha C(\pi, \mathcal{M}) + \beta \sum_{\kappa} I(a_i \in \pi - del) \tag{4}$$

where I is the indicator function indicating if the current action a_i is part of the delete-relaxed plan $\pi - del$ from the current state to each of the landmarks κ. The details[3,4] of the search are provided in Algorithm 1. Notice that the indicator function only comes into play when projection actions are being pushed into the queue, thus biasing the planner towards producing plans that are easier to identify based on the projections. Also note how the cost of actions corresponding to projections in the plan prefix are discounted by a factor γ (to be set depending on how much the designer wants to incentivize projection actions).

5 Projections for Inexplicable Actions

In the previous section, we had focused on dealing with *ambiguity* of intentions during execution of a plan. Now we will deal with *inexplicability* of actions, i.e. how to use projection capabilities to annotate parts of the world so that a plan under execution “makes sense” to the observer.

Illustrative Example. Going back to our block stacking setting, consider a scenario where the human-in-the-loop asks the robot to make a tower of height three with the red block on top (please refer to the attached supplementary video file). Here the optimal plan from the point of view of the observer is likely to be as follows –

```
>> Explicable Plan
>> action :: pick-up green
>> action :: stack green blue
>> action :: pick-up red
>> action :: stack red green
```

[2]In our demonstration, the plan set was composed of the optimal plan (top-1) to the landmarks induced by the all possible words in the domain needed to reach the goal of forming any word.

[3]We currently handle only APs in the solution to a PAPP. Also, the number of APs in a solution were restricted to a maximum of two to three due to the time consuming nature of computing Π. This can be sped up by *precomputing* the relaxed planning graph.

[4]$AP(a) = \langle c, \varnothing, \varnothing \rangle$ refers to the projection action corresponding to action $a \in \mathcal{A}$. Similarly, $AP^{-1}(a)$ denotes the physical action corresponding to a projection action a.

Algorithm 1 Projection-Aware Planning Algorithm

```
1: procedure PAPP-SEARCH
2:   Input: PAPP $\Lambda = \langle \mathcal{M}, \kappa, \{AP\}, \phi \rangle$
3:   Output: Plan $\pi$
4:   Procedure:
5:   $\mathcal{A} \leftarrow \mathcal{A} \cup \{AP\}$                ▷ Add projections to action set
6:   fringe ← Priority_Queue()
7:   fringe.push($\langle \mathcal{I}, \langle\rangle \rangle, 0$)
8:   while True do
9:       $\langle \widehat{S}, \hat{\pi} \rangle, c \leftarrow$ fringe.pop()
10:          if goal check true then return $\hat{\pi}$          ▷ Refer to Section 4.4
11:          else
12:              for $a \in A$ do
13:                  if $\widehat{s} \models pre(a)$ then
14:                      $\widehat{s}' \leftarrow \delta(\widehat{s}, a)$
15:                      fringe.push($\langle \widehat{s}',\ \hat{\pi} + a \rangle,\ F(\widehat{s}', a, \hat{\pi})$)
16: procedure F($\widehat{s}', a, \hat{\pi}$)
17:   if $a \notin \{AP\}$ and $AP(a) \notin \hat{\pi}$ then
18:       return $c_a + C(\hat{\pi}, \mathcal{M})$
19:   else
20:       if $a \in \{AP\}$ then
21:           Compute $\Pi = \{delete - relaxed\ plans\ to\ \kappa\}$
22:           $N \leftarrow 0$
23:           for $\pi \in \Pi$ do
24:               if $AP^{-1}(a) \in \pi$ then
25:                   $N \leftarrow N + 1$
26:           return $\alpha(\ c_a + C(\hat{\pi}, \mathcal{M})\ ) + \beta N$       ▷ (Equation 4)
27:       else
28:           return $\gamma c_a + C(\hat{\pi}, \mathcal{M})$
```

However, not all the blocks are reachable, as determined by the internal trajectory constraints of the robot. So the optimal plan for the robot would instead be longer –

```
>> Robot Optimal Plan
>> action :: pick-up red
>> action :: put-down red
>> action :: pick-up yellow
>> action :: stack yellow blue
>> action :: pick-up red
>> action :: stack red blue
```

This plan is, of course, inexplicable if the observer knows that the robot is a rational agent, given the former’s understanding of the robot model. The robot can chose to mitigate this situation by annotating the unreachable blocks as *“not reachable”* as shown in Figure 6. A video demonstration can be seen at `https://goo.gl/TRZcW6`.

The identification of projection actions in anticipation of inexplicable plans closely follows the notion of multi-model explanations studied in (Chakraborti et al. 2017b).

A Multi-Model Planning Problem is the tuple $\Gamma = \langle \mathcal{M}^R, \mathcal{M}^R_h \rangle$ where $\mathcal{M}^R = \langle D^R, \mathcal{I}^R, \mathcal{G}^R \rangle$ and $\mathcal{M}^R_h = \langle D^R_h, \mathcal{I}^R_h, \mathcal{G}^R_h \rangle$ are respectively the planner’s model of a planning problem and the human’s understanding of it.

In our block stacking domain, multiple models are spawned due to internal constraints of the robot that the human may

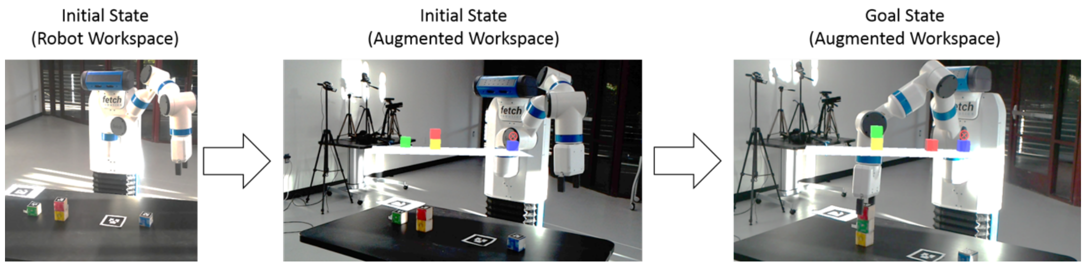


Figure 6: The human has instructed the robot to make a tower of height 3 with the red block on top. Since the blue block is not reachable it has to unstack red in order to achieve its goal. This is a suboptimal plan to the observer who is not aware of the robots internal trajectory constraints. The robot thus decides to project a red error symbol on the blue block indicating it is not reachable. The optimal plans in both models now align.

not be aware of (e.g. reachability) while the world model (i.e. how the world works - the robot has to pick up and object to put it down, etc.) is shared across both the models. As these models diverge, plans that are optimal in the robot's model may no longer be so in the human's and thus become *inexplicable*. The robot can mitigate these situation by generating *multi-model explanations* studied extensively (Chakraborti et al. 2017b; Sreedharan, Chakraborti, and Kambhampati 2017; Chakraborti, Sreedharan, and Kambhampati 2018; Chakraborti et al. 2018a) in recent literature.

A Multi-Model Explanation is a solution to an MMP in the form of a model update to the human so that the optimal plan in the robot's model is now also optimal in the human's updated model. Thus, a solution to Γ involves a plan π and an explanation $\mathcal{E}$ such that –

(1) $C(\pi, \mathcal{M}^R) = C^*_{\mathcal{M}^R}$;

(2) $\widehat{\mathcal{M}}^R_h \longleftarrow \mathcal{M}^R_h + \mathcal{E}$; and

(3) $C(\pi, \widehat{\mathcal{M}}^R_h) = C^*_{\widehat{\mathcal{M}}^R_h}$.

We use the same to generate content for the explanations conveyed succinctly through the medium of mixed reality, as described in the illustrative example above.

6 Conclusion

In conclusion, we showed how an augmented workspace may be used to improve collaboration among humans and robots from the perspective of task planning. This can be either in terms of an interactive plan execution process where the robot can foreshadow future actions to reveal its intentions, or in the context of a *projection-aware* plan generation process where the robot can trade-off the ambiguity in its intentions from the perspective of the human in the loop with the cost of plans. Finally, we showed how explanatory "dialogs" with the human as a response to inexplicable plans can be conducted in this mixed-reality medium.

In recent work (Chakraborti et al. 2018b), we looked at how the beliefs and intentions of an AI agent can be visualized for transparency of its decision-making processes – we refer to this as a process of *"externalization of the brain"* of the agent. Mixed-reality techniques, such as the ones discussed in this paper, can play a pivotal role in this process as we demonstrate in (Sengupta, Chakraborti, and Kambhampati 2018). Indeed, interfacing with virtual agents embody many parallels to gamut of possibilities in human-robot interaction (Williams et al. 2018).

Video Demonstrations. Demonstrations of all the use cases in the paper can be viewed at `https://goo.gl/Gr47h8`. The code base for the projection-aware plan generation and execution algorithms are available at `https://github.com/TathagataChakraborti/ppap`.

Acknowledgements. This research is supported in part by the AFOSR grant FA9550-18-1-0067, the ONR grants N00014-16-1-2892, N00014-13-1-0176, N00014-13-1-0519, N00014-15-1-2027, N00014-18-1-2442 and the NASA grant NNX17AD06G. The first author is also supported by the IBM Ph.D. Fellowship from 2016-18. We also thank Professors Heni Ben Amor and Yu "Tony" Zhang (Arizona State University) for their valuable inputs.

References

Automation World. 2016. President Barack Obama and German Chancellor Angela Merkel visit Weidmullers Industry 4.0 Cockpit. `https://bit.ly/2ICUjD5`.

Linette Lopez. 2018. The robots are killing Tesla. `https://read.bi/2IcOxrw`. Business Insider.

Andersen, R. S.; Madsen, O.; Moeslund, T. B.; and Amor, H. B. 2016. Projecting robot intentions into human environments. In *RO-MAN*, 294–301.

Bryce, D., and Kambhampati, S. A tutorial on planning graph based reachability heuristics. *AI Magazine*.

Chadalavada, R. T.; Andreasson, H.; Krug, R.; and Lilienthal, A. J. 2015. That's on my mind! robot to human intention communication through on-board projection on shared floor space. In *ECMR*.

Chakraborti, T.; Briggs, G.; Talamadupula, K.; Zhang, Y.; Scheutz, M.; Smith, D.; and Kambhampati, S. 2015. Planning for serendipity. In *IROS*, 5300–5306.

Chakraborti, T.; Zhang, Y.; Smith, D.; and Kambhampati., S. 2016. Planning with resource conflicts in human-robot cohabitation. In *AAMAS*.

Chakraborti, T.; Sreedharan, S.; Kulkarni, A.; and Kambhampati, S. 2017a. Alternative modes of interaction in proximal human-in-the-loop operation of robots. *CoRR* abs/1703.08930. UISP 2017 and ICAPS 2017 Demo Track.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017b. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *IJCAI*.

Chakraborti, T.; Talamadupula, K.; Dholakia, M.; Srivastava, B.; Kephart, J. O.; and Bellamy, R. K. 2017c. Mr.Jones – Towards a Proactive Smart Room Orchestrator. In *AAAI Fall Symposium on Human-Agent Groups*.

Chakraborti, T.; Sreedharan, S.; Grover, S.; and Kambhampati, S. 2018a. Plan Explanations as Model Reconciliation – An Empirical Study. *ArXiv e-prints*.

Chakraborti, T.; Fadnis, K. P.; Talamadupula, K.; Dholakia, M.; Srivastava, B.; Kephart, J. O.; and Bellamy, R. K. 2018b. Visualizations for an explainable planning agent. *UISP*.

Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2018. Balancing Explanations and Explicability in Human-Aware Planning. In *AAMAS Extended Abstract*.

Christensen, H. I.; Batzinger, T.; Bekris, K.; Bohringer, K.; Bordogna, J.; Bradski, G.; Brock, O.; Burnstein, J.; Fuhlbrigge, T.; Eastman, R.; et al. 2009. A Roadmap for US Robotics: From Internet to Robotics.

Chrpa, L., and Barták, R. 2009. Reformulating planning problems by eliminating unpromising actions.

Dragan, A., and Srinivasa, S. 2013. Generating legible motion. In *Proceedings of Robotics: Science and Systems*.

Gong, Z., and Zhang, Y. 2018. Robot signaling its intentions in H-R teaming. In *HRI 2018 Workshop on Explainable Robotic Systems*.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*.

International Planning Competition. 2011. IPC Competition Domains. `https://goo.gl/i35bxc`.

Ishii, K.; Zhao, S.; Inami, M.; Igarashi, T.; and Imai, M. 2009. Designing laser gesture interface for robot control. In *INTERACT*.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *IJCAI*.

Karpas, E.; Levine, S. J.; Yu, P.; and Williams, B. C. 2015. Robust execution of plans for human-robot teams. In *ICAPS*.

Kulkarni, A.; Chakraborti, T.; Zha, Y.; Vadlamudi, S. G.; Zhang, Y.; and Kambhampati, S. 2016. Explicable Robot Planning as Minimizing Distance from Expected Behavior. *CoRR* abs/1611.05497.

Leutert, F.; Herrmann, C.; and Schilling, K. 2013. A spatial augmented reality system for intuitive display of robotic data. In *HRI*.

Look, C. 2016. Robots are coming, but not for your job. Bloomberg. `https://goo.gl/ePzcta`.

MacNally, A. M.; Lipovetzky, N.; Ramirez, M.; and Pearce, A. R. 2018. Action selection for transparent planning. *AAMAS*.

Maurtua, I.; Pedrocchi, N.; Orlandini, A.; de Gea Fernández, J.; Vogel, C.; Geenen, A.; Althoefer, K.; and Shafti, A. 2016. Fourbythree: Imagine humans and robots working hand in hand. In *ETFA*, 1–8. IEEE.

Mistry, P.; Ishii, K.; Inami, M.; and Igarashi, T. 2010. Blinkbot: look at, blink and move. In *UIST*, 397–398. ACM.

Muro, M., and Andes, S. 2015. Robots seem to be improving productivity, not costing jobs. Harvard Business Review.

Omidshafiei, S.; Agha-Mohammadi, A.-A.; Chen, Y. F.; Ure, N. K.; How, J. P.; Vian, J.; and Surati, R. 2015. Mar-cps: Measurable augmented reality for prototyping cyber-physical systems. In *AIAA ARC*.

Omidshafiei, S.; Agha-Mohammadi, A.-A.; Chen, Y. F.; Ure, N. K.; Liu, S.-Y.; Lopez, B. T.; Surati, R.; How, J. P.; and Vian, J. 2016. Measurable augmented reality for prototyping cyberphysical systems: A robotics platform to aid the hardware prototyping and performance testing of algorithms. *IEEE Control Systems* 36(6):65–87.

PRNewswire. 2016. Global survey: 1.3 million industrial robots to enter service by 2018. IFR.

Ramırez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *AAAI*.

Riabov, A.; Sohrabi, S.; and Udrea, O. 2014. New algorithms for the top-k planning problem. In *SPARK*, 10–16.

Robotenomics. 2017. Industrial robot.

Ruffaldi, E.; Brizzi, F.; Tecchia, F.; and Bacinelli, S. 2016. Third point of view augmented reality for robot intentions visualization. In *AVR*, 471–478. Springer.

Sato, S., and Sakane, S. 2000. A human-robot interface using an interactive hand pointer that projects a mark in the real work space. In *ICRA*, volume 1, 589–595. IEEE.

Sengupta, S.; Chakraborti, T.; and Kambhampati, S. 2018. MA-RADAR – A Mixed-Reality Interface for Collaborative Decision Making. *UISP*.

Shen, J.; Jin, J.; and Gans, N. 2013. A multi-view camera-projector system for object detection and robot-human feedback. In *ICRA*.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2017. Handling model uncertainty and multiplicity in explanations via model reconciliation. In *ICAPS*.

Tellex, S.; Knepper, R.; Li, A.; Rus, D.; and Roy, N. 2014. Asking for help using inverse semantics. In *RSS*.

Turk, M., and Fragoso, V. 2015. Computer vision for mobile augmented reality. In *Mobile Cloud Visual Media Computing: From Interaction to Service*.

Watanabe, A.; Ikeda, T.; Morales, Y.; Shinozawa, K.; Miyashita, T.; and Hagita, N. 2015. Communicating robotic navigational intentions. In *IROS*, 5763–5769. IEEE.

Weiss, S. 2015. Could your robot hurt you? perhaps, but not intentionally. Brown Human-Robot Interaction (HRI) Lab.

Williams, T.; Szafir, D.; Chakraborti, T.; and Ben Amor, H. 2018. Virtual, augmented, and mixed reality for human-robot interaction. In *Companion of the International Conference on Human-Robot Interaction*, 403–404. ACM.

Zhang, Y.; Sreedharan, S.; Kulkarni, A.; Chakraborti, T.; Zhuo, H. H.; and Kambhampati, S. 2017. Plan Explicability and Predictability for Robot Task Planning. In *ICRA*.

NL2PDDL: A Conversational Interface for Model Generation and Iteration

Kshitij P. Fadnis and **Kartik Talamadupula**
IBM Research
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{kpfadnis, krtalamad} @ us.ibm.com

Abstract

Although the automated planning community has seen many advances to planning techniques in the past decade, domain model creation and maintenance has remained the central bottleneck preventing wider adoption of planning technology in the real world. While there has been some work on learning these models in an automated fashion, there has been very little focus on user-friendly interfaces for the creation, querying, and editing of planning models. In this work, we present a novel approach to interfacing with planning models using natural language via a conversation modality. We detail the construction of the system, its capabilities, as well as some key opportunities and challenges that are brought up by this unique interface with planning models.

1 Introduction

In recent years, automated planning techniques and systems have achieved an impressive scale-up in terms of the size of the problems that they can handle. Planners such as Fast Downward (Helmert 2006) and its descendants routinely solve problem instances of real world size in a matter of seconds. Unfortunately, this scale-up has not led to a significant increase in the adoption of planning techniques for real world applications. One of the main reasons for this is the extremely cumbersome task of specifying domain models for planning tasks. The most widely accepted specification language – PDDL (Mcdermott et al. 1998) – and its variants are still fairly complex and have a steep learning curve. This complexity of PDDL is a necessary evil, since the language also needs to be expressive enough to model real domains. Indeed, much planning work over the past two decades has focused on this tension between increased expressivity on the one hand, and planner efficiency on the other.

Rather than focus on this known problem, planning practitioners should instead look at the other, much narrower end of this real world bottleneck for planners – the modeling and specification of domains. Even today, this is more of a dark art than a science, and it is handled by a small group that can afford to spend the time needed to truly straddle two realms. One is the world of planning research; the other is that of the real world application. Only such people are currently able to identify the correct abstraction level needed to convert an application into a decision making problem, and then model that abstraction into PDDL. This has long remained the elephant in the room for the planning community: an inconvenient truth that gets truly discussed and debated only in workshops and systems demonstrations.

In this paper, we introduce the `NL2PDDL` system, whose ultimate form is intended to make the specification of planning domains – agnostic of representation language – easier for subject matter experts (SMEs) who are not familiar with planning technology. Our system introduces, for the first time, the medium of conversation (and dialog) as the backing technology for the creation and maintenance of planning models. The reasons for using this medium are straightforward, and we list them here.

Conversation as a Medium for Planning One of the main reasons for using conversation is that expressing new ideas and specifications is easier through natural language, and certainly more natural than filling out cumbersome forms based on the rules of a syntax. Besides, while using conversation, the burden of translating an utterance into the representation language's syntax falls on the authoring system, and not on the SMEs who are using the system. This lends itself to easy automation of the translation from natural language to the desired representation, while ensuring that the cognitive burden of learning yet another new interface is non-existent. Conversation also provides a natural avenue for two-way – and if needed, multi-way – interaction. The system can prompt, clarify, and inform the user when necessary; this can even be done without explicit tasking, as the agent recognizes mistakes being made in the authoring. Such an interactive process – conducted through dialog – is also very educational, and can be used to teach the process and pitfalls of domain modeling to an SME who is a novice when it comes to PDDL.

Finally, the use of conversation gives the system a stored history of the domain authoring process. While version control systems can track model updates just as well, there is a fundamental difference in terms of the information that is stored: version control techniques are restricted to the syntactic information alone, while the conversation history can capture the SME's interaction in its entirety. A good domain modeling assistant can use this rich history from past conversations to ensure that the modeling process is on the right track. The conversation thus serves as a natural, human-readable log of the modeling process: this can be used when the domain needs to be augmented, or a similar

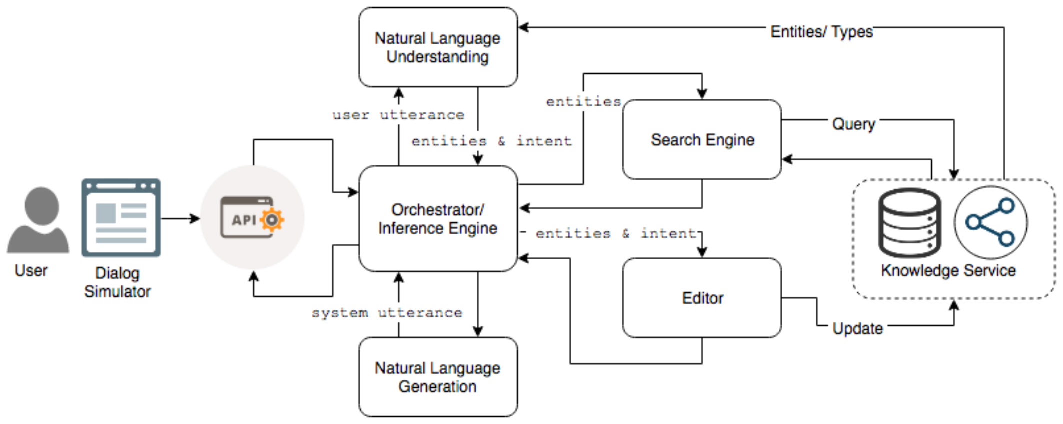


Figure 1: The architecture of the NL2PDDL system.

domain modeled sometime in the future. The conversation history is also a valuable resource to train language models for better prompt generation and interaction.

In this paper, we detail the first steps towards realizing a conversational assistant for domain modeling tasks via the NL2PDDL system. We first look at related work in the field of knowledge engineering for planning and scheduling (KEPS), and point out differences and synergies with our contribution. We then describe the architecture of the NL2PDDL system and its constituent components, as well as a specific web-based implementation of a minimum viable product. This is followed by a demonstration of the system's current capabilities, and a discussion about existing limitations and future work. This paper describes the first iteration of an ambitious system, and we ask that it be seen as a report on work in progress.

2 Related Work

Over the past decade or so, the planning community has embraced the difficulty as well as the importance of the knowledge engineering task; and the bottleneck posed by it for the wider adoption of planning technology. Motivated by McCluskey's early work (McCluskey 2000) and the KEPS (Knowledge Engineering in Planning and Scheduling) series of workshops, this area has received steady attention.

The prior work we highlight in this section can be divided into three distinct categories: theoretical, applications, and tools. Theoretical work focuses on the verification and modeling challenges inherent in the knowledge engineering for planning space, without looking at specific application domains. Application work, on the other hand, combines planning techniques with real world problem domains. Finally, tools offer an implementation of knowledge engineering methods in an overall system, in order to elicit planning models. Our work intersects with all three of these categories, and can be extended in conjunction with many of the prior efforts mentioned here.

Theoretical The backing engine for a tool like NL2PDDL needs to necessarily accommodate functionalities that determine whether a planning model is being built correctly, and in a fashion that will work efficiently with planners. Although the current beta version of our tool does not handle this – and instead places the burden of ensuring model correctness on the user – prior work on testing and verifying that a model satisfies a given set of requirements (Raimondi, Pecheur, and Brat 2009) can be very useful for this extension. Another important issue that needs to be handled is that of domain model configuration (Vallati et al. 2015) – this will eventually determine the computational efficiency of planners that use such a model. Finally, more recent work by McCluskey et al. (McCluskey, Vaquero, and Vallati 2016; 2017) defines measures for planning model elicitation that go beyond mere correctness, and look instead at a host of other metrics including accuracy, consistency, completeness, adequacy, and operationality. All of these ideas can be operationalized in a framework such as VAL (Howey, Long, and Fox 2004), which can then be integrated with NL2PDDL.

Applications Natural language interfaces – of the kind used by the NL2PDDL system – have not been used extensively in conjunction with automated planning work. To be sure, there has been a lot of work in the opposite direction, in applying planning techniques to the problem of dialog management: led by the seminal work of Williams & Young (2007). However, to the best of our knowledge, the work by Yates et al. (Yates, Etzioni, and Weld 2003) is the only work that uses natural language as a medium to automatically generate planning knowledge – in their case, the goals needed for household appliances to carry out their tasks. We believe there is much more promise in this direction. On the model creation and maintenance side, a rather interesting application of planning was the work of Puissant et al. (Puissant, Mens, and Van Der Straeten 2010) on resolving model inconsistencies in software engineering models with the help of automated planning.

Tools Knowledge engineering have vastly improved in the past decade, under the aegis of the twin KEPS workshops and the International Competition on KEPS (ICKEPS). Most modeling tools build up from some notion of ontology all the way to the planning model (Vaquero et al. 2013; Wickler, Chrpa, and McCluskey 2014); while PDDL

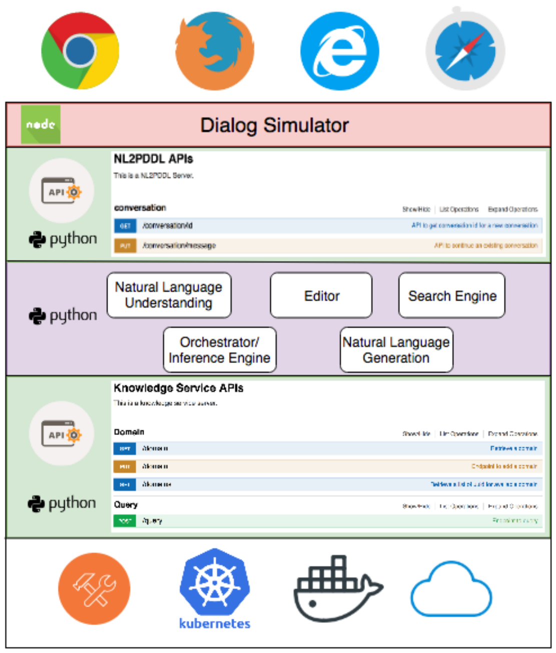

Dialog Simulator
NL2PDDL APIs
python
Natural Language Understanding
Editor
Search Engine
Orchestrator/ Inference Engine
Natural Language Generation
Knowledge Service APIs
kubernetes

Figure 2: The `NL2PDDL` system's infrastructure.

`Studio` (Plch et al. 2012) provides a number of useful features for users who are already familiar with PDDL. A good overview of such tools is provided by Jilani et al. (Jilani et al. 2014). PRIDE (Bonasso and Boddy 2010) supports a complex query structure, but relies heavily on the availability of a domain ontology: a requirement that is self-identified by the authors as non-trivial. PRONTOE (Bell et al. 2013) addresses ontology design by providing a graphical editing tool that allows users to define various ontology kernels that can be used to separate complex systems into their constituent components – however, it is accompanied by a steep learning curve when it comes to usage. itSIMPLE$_{2.0}$ (Vaquero et al. 2013) similarly uses a UML-driven interface for domain modeling, which comes with its own learning curve and limitations. Our work is complementary to many of these tools, in that it can be extended by combining with any (or indeed all) of them. However, the availability of a simple natural language interface will significantly decrease the load on users trying to model novel domains.

The work that is closest in spirit to the `NL2PDDL` system is that of Sethia et al. (Sethia, Talamadupula, and Kambhampati 2014), which uses the Google API in order to construct a mapping between higher level actions at the PDDL abstraction, and lower level actions that are present in a humanoid robot's internal libraries. However, even this interface is very form-like, and insists upon the user providing information in the same order as that desired by the PDDL syntax. It also does not allow the user to query an existing model.

3 System Architecture

In this section, we describe the architecture of the `NL2PDDL` system, outlined in Figure 1. We follow a modular design, where each component of the system can be accessed through a simplified set of APIs. There are six major components in the system:

- **Natural Language Understanding (NLU)**: The job of the NLU component is to process a text utterance, in order to identify the entities and intents present in it. For example, on the input "show me some actions in this domain", the output will be the entity "actions" and the intent "inform". We employ lexical and fuzzy matching to explore a given input for valid entities, while intent is identified using the Watson Natural Language Classifier (NLC) by IBM. The NLC is trained with ten different intents, which broadly cover the space of interaction this system is designed to handle.[1] One of the key features of our NLU is that it restricts its entity matching to a set of concepts and their paraphrases from the domain of interest. This reduces the scope of the entity detection, which in turn provides both speed-up and an accuracy boost.
- **Search Engine**: The search engine component is primarily responsible for retrieving knowledge about the entities that are present in the text input. This component formulates queries based on the available entities and their subfields.
- **Knowledge Service**: This service is the internal memory of the system. It reads in an existing planning domain (when available) and builds a graphical representation of the concepts in that domain. Each type, predicate, and action are considered a separate node in this graph; the edges reflect the associations between them. For example, an action `X` with precondition `y` will feature an edge from the `x` node to the `y` node, with the "precondition" relation. This graphical representation allows for easier query and update mechanisms over planning domains. The knowledge service also bootstraps the NLU with concepts associated with the domain of interest.
- **Editor**: The Editor is responsible for edits to the model that are requested by a user. These include things like the addition or removal of type definitions, the addition or removal of predicates from the add or delete effects of an action, and so on. The Editor retrieves the entities identified by the NLU and sends the appropriate update request to the Knowledge Service. The Editor is currently intended to support simple edit capabilities like the removal of types, predicates, or actions by name; and the addition of new types. We are in the process of extending the Editor's functionality to handle more complex operations like action editing.
- **Natural Language Generation (NLG)**: The NLG is a critical component that differentiates `NL2PDDL` from previous model editors, with its ability to produce conversation. It enables feedback in natural language, which makes interaction with the system more natural. We currently employ a rule-based system with a few predefined response classes; however, our aim is to replace this with

[1]The data that we used to train this classifier will be made available, along with the entire source code.

a generative model in order to deliver more human-like responses.

- **Orechestrator/Inference Engine:** This component is the brain of the system – all communication flows through it. Messages in the system are tagged with their source at the point of their origin. The job of the orchestrator is to determine where the message should be sent next, based on where it came from and what it contains. For example, a message originating from the Search Engine is tagged with that source; when it arrives at the orchestrator, it is redirected to the NLG in order to generate an appropriate response based on the search results.

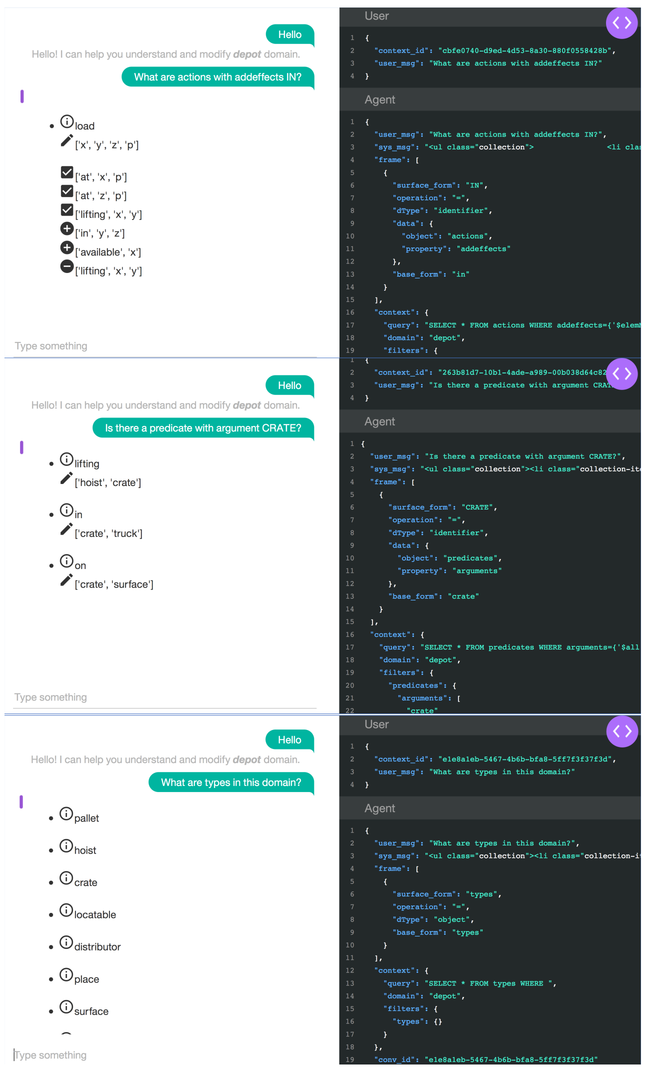
On the platform side, the knowledge service is a standalone RESTful service deployed as a Docker container in a Kubernetes cluster. All the information serviced by the system is backed up using a persistent storage mechanism. As seen in Figure 2, the Knowledge Service provides the '/domain' and '/query' endpoints. The remaining components of the system (detailed above) are written in Python. `NL2PDDL` also comes with a built-in web-based chat application written in NodeJS; this is introduced in the next section.

4 Instruction Set

As we have stated earlier in the paper, this is a first-of-a-kind solution that attempts to provide a natural language interface via the conversation modality for planning models. We use this section to advise readers on the current (limited) expressiveness of the NLU component. The NLU handles spelling mistakes in the name of concepts (types, predicates, actions) as well as certain grammatical mistakes; this flexibility exists as long as syntactic parsing and part-of-speech tagging are still possible on the input sentence. To assist readers, we list a few typical queries that are currently supported by the system:

- **What-type questions** regarding top-level concepts:
  - `What are types available in this domain?`
  - `What are actions in this domain?`
  - `What are available predicates?`
- **What-type** and **To be-type questions** with a condition:[2]
  - `Is there action named <NAME>?`
  - `What are actions with addeffects <PREDICATE-NAME>?`
  - `Is there a predicate with argument <TYPE-NAME>?`
  - `What are all actions with preconditions <PREDICATE-NAME>?`
- **Greeting statements**: Simple greetings to initiate the conversation.
  - `hello`
  - `hi`
  - `good morning`

[2]Please note that words like `named`, `addeffects` and `argument` are necessary for conditional statements in order to uniquely identify appropriate nested concepts. Proper nouns like action, type, or predicate names must also be capitalized.


Hello

Hello! I can help you understand and modify *depot* domain.

What are actions with addeffects IN?

- load
  ['x', 'y', 'z', 'p']
  ['at', 'x', 'p']
  ['at', 'z', 'p']
  ['lifting', 'x', 'y']
  ['in', 'y', 'z']
  ['available', 'x']
  ['lifting', 'x', 'y']

Type something

User
```
{
  "context_id": "cbfe0740-d9ed-4d53-8a30-880f0558428b",
  "user_msg": "What are actions with addeffects IN?"
}
```
Agent
```
{
  "user_msg": "What are actions with addeffects IN?",
  "sys_msg": "<ul class="collection">                <li clas
  "frame": [
    {
      "surface_form": "IN",
      "operation": "=",
      "dType": "identifier",
      "data": {
        "object": "actions",
        "property": "addeffects"
      },
      "base_form": "in"
    }
  ],
  "context": {
    "query": "SELECT * FROM actions WHERE addeffects={'$elem
    "domain": "depot",
    "filters": {
```

Hello

Hello! I can help you understand and modify *depot* domain.

Is there a predicate with argument CRATE?

- lifting
  ['hoist', 'crate']
- in
  ['crate', 'truck']
- on
  ['crate', 'surface']

Type something

```
{
  "context_id": "263b81d7-10b1-4ade-a989-00b038d64c82
  "user_msg": "Is there a predicate with argument CRA
}
```
Agent
```
{
 "user_msg": "Is there a predicate with argument CRATE?",
 "sys_msg": "<ul class="collection"><li class="collection-ite
 "frame": [
   {
     "surface_form": "CRATE",
     "operation": "=",
     "dType": "identifier",
     "data": {
       "object": "predicates",
       "property": "arguments"
     },
     "base_form": "crate"
   }
 ],
 "context": {
   "query": "SELECT * FROM predicates WHERE arguments={'$all
   "domain": "depot",
   "filters": {
     "predicates": {
       "arguments": [
         "crate"
```

Hello

Hello! I can help you understand and modify *depot* domain.

What are types in this domain?

- pallet
- hoist
- crate
- locatable
- distributor
- place
- surface

Type something

User
```
{
  "context_id": "e1e8a1eb-5467-4b6b-bfa8-5ff7f3f37f3d",
  "user_msg": "What are types in this domain?"
}
```
Agent
```
{
  "user_msg": "What are types in this domain?",
  "sys_msg": "<ul class="collection"><li class="collection-i
  "frame": [
    {
      "surface_form": "types",
      "operation": "=",
      "dType": "object",
      "base_form": "types"
    }
  ],
  "context": {
    "query": "SELECT * FROM types WHERE ",
    "domain": "depot",
    "filters": {
      "types": {}
    }
  },
  "conv_id": "e1e8a1eb-5467-4b6b-bfa8-5ff7f3f37f3d"
```

Figure 3: Some examples of queries that are possible on `NL2PDDL`: (a) Action (b) Predicate (c) Types.

  - `good evening`
- **Closing statements**: Simple utterances to end the current conversation.
  - `thank you`
  - `no. that will be it.`
  - `i am all set`
  - `nm. i am done.`
  - `thanks`

5 Demonstration

The `NL2PDDL` system currently includes a web-based interface, which can be used to interact with the system and explore a planning model. A live demonstration of our system and this interface can be viewed at the following URL: `ibm.biz/nl2pddl`. Please see Section 4 for a description of the interactions that are currently supported.

In this section, we explain the various features of this interface via a few examples. In Figure 3(a), we ask the system to return actions that satisfy a given query pertaining to add effects. The system returns an action ('load') that satisfies this query. The right-hand side pane on the interface allows a user to check on how the system interpreted their utterance. In this particular example, the system identified that an action with an add effect on the 'in' predicate was being requested. Figure 3(b) and Figure 3(c) demonstrate that we currently support different degrees of granularity when inquiring about the domain physics. Figure 3(c) highlights the system's ability to look for major artifacts in the domain definition, like types, predicates, or actions. Figure 3(b), on the other hand, shows an example of narrowing the scope of the query further, in order to identify predicates that take certain argument types.

The system allows for spelling mistakes by using a built-in spelling correction mechanism with a settable parameter for its minimum match threshold. Currently, we set it to require a minimum threshold of 85% spelling match with a suggested correction. Each intent also features eight to ten phrasal variations, granting coverage over a wider range of conversation. Finally, the system currently only supports a limited subset of possible ways to declare an entity. Specifically, we insist that users capitalize the name of an action, predicate, or type fully: for example, 'IN' in Figure 3(a), or 'CRATE' in Figure 3(b).

6 Future Work

As we have stressed throughout this paper, the NL2PDDL system is currently the first, work-in-progress version of a much larger vision that involves more natural authoring and maintenance of planning models. There are many avenues of extension, which we list here in no specific order. First, we have partially implemented an 'edit' functionality that complements the 'query' functionality that we demonstrated in this paper. This allows users to add or change things in the domain of interest; we will be deploying this shortly. We are also extending the system so that it can deal with more than just a single proof-of-concept domain (currently Depots). A more challenging future direction involves the integration of NL2PDDL with existing domain authoring, maintenance, and verification tools like VAL and itSIMPLE: we would like for the system to provide (automated) intelligent feedback to domain authors. On the natural language and conversation side, we are exploring ways to liberate the system from its current rule-based NLU and NLG implementations, and move to a more natural, automatically learned model of conversation. Finally, we hope to release a more polished interface for demonstration at ICAPS 2018.

References

Bell, S.; Bonasso, P.; Boddy, M.; Kortenkamp, D.; and Schreckenghost, D. 2013. Prontoe a case study for developing ontologies for operations.

Bonasso, P., and Boddy, M. 2010. Eliciting planning information from subject matter experts. In *Proceedings of ICAPS 2010 Workshop on Scheduling and Knowledge Engineering for Planning and Scheduling (KEPS)*, 5–12.

Helmert, M. 2006. The fast downward planning system. *J. Artif. Int. Res.* 26(1):191–246.

Howey, R.; Long, D.; and Fox, M. 2004. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, 294–301. IEEE.

Jilani, R.; Crampton, A.; Kitchin, D. E.; and Vallati, M. 2014. Automated knowledge engineering tools in planning: state-of-the-art and future challenges.

McCluskey, T. L.; Vaquero, T.; and Vallati, M. 2016. Issues in planning domain model engineering.

McCluskey, T. L.; Vaquero, T. S.; and Vallati, M. 2017. Engineering knowledge for automated planning: Towards a notion of quality. In *Proceedings of the Knowledge Capture Conference*, 14. ACM.

McCluskey, T. 2000. Knowledge engineering for planning roadmap.

Mcdermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL - the planning domain definition language. Technical Report TR-98-003, Yale Center for Computational Vision and Control,.

Plch, T.; Chomut, M.; Brom, C.; and Barták, R. 2012. Inspect, edit and debug pddl documents: Simply and efficiently with pddl studio. *System Demonstrations and Exhibits at ICAPS* 15–18.

Puissant, J. P.; Mens, T.; and Van Der Straeten, R. 2010. Resolving model inconsistencies with automated planning. In *Proceedings of the 3rd Workshop on Living with Inconsistencies in Software Development*, 8–14.

Raimondi, F.; Pecheur, C.; and Brat, G. 2009. Pdver, a tool to verify pddl planning domains.

Sethia, S.; Talamadupula, K.; and Kambhampati, S. 2014. Teach Me How To Work: Natural Language Model Updates and Action Sequencing. In *ICAPS 2014 Systems Demonstrations*.

Vallati, M.; Hutter, F.; Chrpa, L.; and McCluskey, T. L. 2015. On the effective configuration of planning domain models. In *IJCAI*, 1704–1711.

Vaquero, T. S.; Silva, J. R.; Tonidandel, F.; and Beck, J. C. 2013. itsimple: towards an integrated design system for real planning applications. *The Knowledge Engineering Review* 28(2):215–230.

Wickler, G.; Chrpa, L.; and McCluskey, T. L. 2014. Kewi: A knowledge engineering tool for modelling ai planning tasks.

Williams, J. D., and Young, S. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language* 21(2):393–422.

Yates, A.; Etzioni, O.; and Weld, D. 2003. A reliable natural language interface to household appliances. In *Proceedings of the 8th international conference on Intelligent user interfaces*, 189–196. ACM.

Generating Human Work Instructions from Assembly Plans

Csaba Kardos[a,b], András Kovács[a], Balázs E. Pataki[c], József Váncza[a,b]
[a]EPIC Center of Excellence in Production Informatics and Control, Inst. Comp. Sci. & Control, Hun. Acad. Sci.
[b]Department of Manufacturing Science and Technology, Budapest University of Technology and Economics,
[c]Department of Distributed Systems, Institute for Computer Science and Control, Hungarian Academy of Sciences,
{csaba.kardos, andras.kovacs, pataki.balazs, vancza}@sztaki.mta.hu

Abstract

Despite enormous robotization efforts, most of the assembly process is still executed by human workforce in many industries performing the assembly of mechanical products. Therefore, a crucial component of any automated planning system in those applications is the worker instruction system that presents the automatically generated plans to human assembly workers. In case of complex products and processes, finding the most efficient presentation to workers with different skills and background is a great challenge. This paper proposes novel methods for generating context-dependent, animated work instructions from automatically generated assembly plans. The proposed approach is demonstrated on an industrial case study that involves the manual assembly of an automotive supercharger.

Introduction

While a significant portion of the research on automated planning and scheduling focuses on decision making in autonomous systems, in many applications, the generated plans are still executed by human actors. In such applications, a key success factor is the effective instruction system that presents the complex plan to the human worker. This is the case in mechanical assembly as well: despite intensive robotization efforts, the assembly of mechanical products in many industries still requires human workforce.

The motivation for replacing classical static, often still paper-based work instructions by enhanced digital Worker Instruction Systems (WIS) and automatically generated instructions is manifold. Firstly, shortening product life cycles and increasing variety make it difficult to maintain and regularly update static instructions, and hence, the ability to dynamically adjust the digital content to the changing production environment is the most attractive characteristic of WIS (Leu et al. 2013; Lušić et al. 2014). Most commercial WIS implementations offer interfaces to Enterprise Resource Planning (ERP), Product Lifecycle Management (PLM) and Manufacturing Execution Systems (MES), which enables the import of product and process data, supports the definition of instruction templates, and hence, allows maintaining consistent and up-to-date work instructions.

In line with the above, vast effort is being invested into *computer-aided process planning* (CAPP) techniques to manage the complexity of planning, and to support the process engineer in defining the most efficient assembly processes for the products (Hu et al. 2011; Ghandi and Masehian 2015). As the final step of the process planning workflow is the generation of program codes for robotic resources and instructions for human workers, the automation of process planning is incomplete without automated methods for instruction generation as well (Kaipa et al. 2012).

Finally, it must be recognized that the increasing complexity of products and production systems puts a heavy cognitive load on assembly workers as well (Leu et al. 2013; Lušić et al. 2016). Therefore, human workers need enhanced support from the WIS, in the form of unambiguous instructions delivered using the modality that suits the given environment the best; in addition to classical text instructions, figures, videos, 3D animations, audio instructions, or even augmented reality (AR) can be used, too. Instructions can be tailored further by exploiting context awareness (Bader and Aehnelt 2014; Bannat et al. 2008; Claeys et al. 2016). Efficient WIS and instructions have a special role in supporting the training of the workers (Michalos et al. 2014) and in solving one-of-a-kind issues in maintenance and rework operations (Fiorentino et al. 2014).

This paper presents novel methods for the automated generation of work instructions for mechanical assembly. The methods proposed in this paper support the final step of the workflow in a full-fledged workcell configuration toolbox, which includes efficient optimization methods for assembly sequence planning, resource assignment (i.e., action planning) and path planning, by presenting the resulting plans to human workers. For fully exploiting the opportunities of digital WIS, the methods support the generation of context-aware instructions augmented with 3D animations of the assembly process.

Problem Definition

The objective of *assembly workcell configuration* is to manage the journey of assembled products on their way from design ideas to the reality of production. Its two main subproblems, which approach the same problem from the viewpoints of the assembly process and the assembly resources, respectively, are *assembly process planning* and *workcell layout design*. Both of these sub-problems involve various

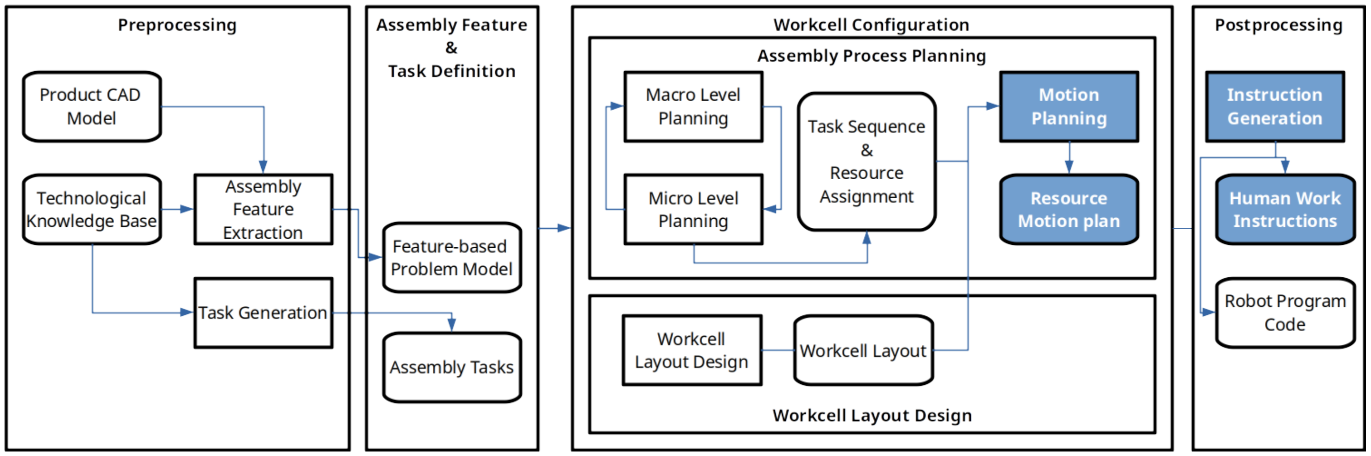


Figure 1: Workflow of assembly workcell process planning. The sub-problem in scope, instruction generation, is highlighted in blue.

types of decisions. In case of assembly process planning, it is common to differentiate *macro-level planning*, which is a combinatorial problem involving decisions on the assembly task sequence and the resources assigned to the *tasks*; and micro-level planning, which is responsible for elaborating the details of each individual task. The latter includes, e.g., assembly path planning, fixturing, or tooling. The workflow of assembly workcell configuration applied in this paper, departing from the conversion of legacy CAD models of the product into feature-based models directly processable for automated planners, and concluding at the generation of human work instructions and robot program codes, is presented in Fig. 1.

In case of assembly tasks performed by human workers, the final step of the workcell configuration workflow is the generation of *work instructions* for the workers, which is the focus of this paper. Instructions must be generated from a structured representation of the assembly process plan, which includes the sequence of assembly tasks, a feature-based representation of the technological content of each task, as well as a collision-free assembly path. It has to be noted that in an actual manufacturing environment, human supervision and interaction are still inevitable before finalizing instructions. Therefore, there are two main user roles in an instruction generation system: process engineers who participate in the creation of instructions, and workers to whom the instructions are delivered via the multi-modal interfaces. The goal here is to automatically generate draft instructions that will be verified and finalized by a process engineer. Thus, effective work instructions must fulfill the following key requirements:

- Instructions must be easily and unambiguously interpretable by the workers. For this purpose, classical textual instructions can be accompanied by images or animations.
- Context-dependent instructions should be applied to fit the actual conditions, e.g., the worker's skill, language, devices and presentation preferences, as much as possible.
- Instructions should be delivered using the modality that suits the actual application the best. Beyond instructions displayed on screens, audio instructions are also in scope.
- The generation of instruction should be automated as much as possible. Nevertheless, the system should enable editing and approving the instructions by a process engineer.
- It should be possible to generate the instructions from legacy representations, such as the existing CAD models of the products.

In the sequel, a brief overview is given of the overall methodology applied to assembly workcell configuration, focusing on sub-problems related to assembly process planning, and then the proposed approach to work instruction generation is presented in detail.

Assembly Planning Approach

Feature-based Assembly Model

A key challenge in process planning in any manufacturing domain is to match the different views of the planning problem, related to geometry, technology, tooling, etc. The classical approach to responding to this challenge is to use a so-called *feature-based* representation, in which features give a holistic description of the micro-worlds related to an individual manufacturing or assembly operation. Examples of features in assembly include *placing* two parts on each other, *insertion* of a part into a hole on another part, or *screwing*. The complete specification of such a feature then consists of the geometries of the corresponding parts, the relative position of the parts in the target configuration, and the definition of the movement that can join the parts, including the direction of the motion and the technological parameters. The model also allows organizing individual features that belong together in a sense, e.g., parallel screwing features that join the same parts, into so-called *composite features*. The assembly planner then assigns a single assembly task to the composite feature, which involves creating all in-

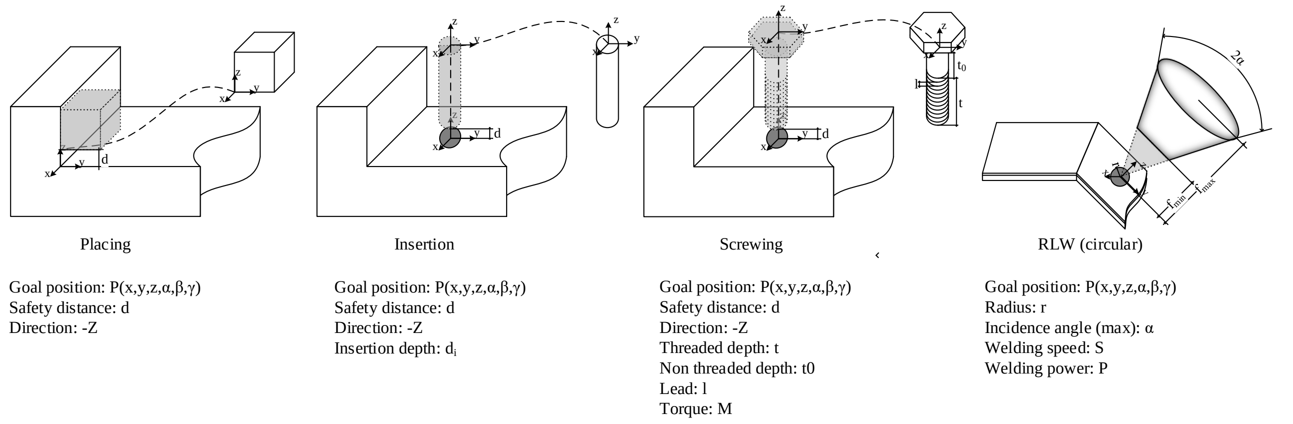


Figure 2: Types of assembly features and their parameters.

dividual features in parallel or in a given sequence, as defined by the ordering flag in the composite (see details later). Some assembly features and their parameters are presented in Fig. 2, whereas a generic presentation of feature-based assembly planning is given in (Wang, Keshavarzmanesh, and Feng 2011).

Formally, there is given a set of assembly tasks (and potentially some auxiliary tasks) that must be executed in an appropriate order using suitable resources to arrive from the set of parts into an assembled product. The technological content of each assembly task is specified by the assembly feature assigned to it, which defines the set of base parts and moved parts, along with the parameters of the motion that joins the two sets of parts (Fig. 2). This motion can be subdivided into two phases: an *approach* motion that takes the parts from their storage locations to a so-called *near* position using an arbitrary collision-free trajectory, and the *local* motion from the near position into the final configuration. The latter, local motion is perfectly defined in the assembly feature. It is assumed that each assembly task requires two resources for its execution: a fixture that holds the base parts at their place and a tool that executes the task on the moved parts. The assembly planner also adds so-called changeover tasks to the plan at points where some of the assigned resources differ between consecutive assembly tasks. The details of the feature-based assembly process planning model are presented in (Kardos, Kovács, and Váncza 2016; 2017).

Assembly Sequence Planning and Resource Assignment

Since assembly process planning involves making diverse types of decisions, it is typical to solve it using decomposition into macro-level planning, which is a combinatorial problem responsible for defining the structure of the plan, and micro-level planning, which elaborates the details of each individual task. In this paper, the decomposition approach presented in (Kardos, Kovács, and Váncza 2017) is adopted, which uses a macro-level planner to optimize the assembly task sequence and the assignment of the resources (tools and fixtures) to the tasks, while a collection of micro-level sub-problem solvers ensures that the planned tasks can be implemented in physical reality. Sub-problem solvers in the current implementation include technological feasibility, collision avoidance, fixturing, and tooling modules.

The solution approach proposed in (Kardos, Kovács, and Váncza 2017) is Benders decomposition with a mixed-integer linear programming (MILP) model applied to solving the macro-level planning master problem and various customized lower-level solvers generating feasibility cuts for the master problem. This approach guarantees the optimality of the macro-level plan and its micro-level feasibility according to all micro-level aspects investigated. The MILP formulation of the master problem is solved using the commercial mathematical programming software FICO Xpress 7.8. The Benders framework, as well as the sub-problem solvers responsible for technology, fixturing, tooling, and collision avoidance, have been implemented in Python. The latter module performs collision detection using the Flexible Collision Library (FCL) (Pan, Chitta, and Manocha 2012) on the STL triangle mesh models of the involved objects.

Assembly Path Planning

Assembly path planning is performed for each task separately at two distinct phases of the workflow: (1) during assembly process planning as a micro-level sub-problem to assess the geometrical feasibility of an assembly sequence, and (2) during motion planning for generating the motion sequence to be actually executed. In both phases, the motion path of the ensemble of moved parts and the attached tool (gripper or an actual tool, e.g., screwdriver), treated as a single solid object, is planned from a remote position to the *near* position of the corresponding assembly feature. The base parts and the fixture are considered to be the obstacles. Planning is performed using an implementation of the classical Rapidly-exploring Random Trees (RRT) algorithm.

In phase (1), when the workcell layout is unknown and the actual resources to carry out the task are not selected yet, path planning handles the moved parts as free-flying objects with 3 or 6 degrees of freedom (DoF): for most of the assemblies investigated during preliminary experiments, it was found that allowing translational movements results in a feasible path when there exists one. Nevertheless, the path planner can consider 6-DoF problems as well for more complicated geometries. The only answer expected from the path planner in this phase is a yes (a collision-free path has been found) or no (the iteration limit has been hit without finding a collision-free path) answer, and the quality of the path is disregarded.

In contrast, at the phase of motion planning, the goal is to generate paths that will be actually executed. Here, the details of the method differ for robots and for humans. For tasks performed by humans, which is the focus of the present paper, path planning is still performed for free-flying objects (hence, it is implicitly assumed that where parts and tools fit, there the human worker can also access the location of the task). However, path smoothing is applied to the results of RRT to arrive at a path which looks reasonable to workers on animated work instructions. For a robotic task, path planning is performed in the robot's joint space, taking into account the results of workcell layout design, including the pick-up and put-away locations of the parts and equipment (the latter function is not fully implemented yet).

Generating and Presenting Work Instructions

Assembly systems are moving towards using digital technologies for handling and dispatching work instructions. Digital WISs offer, on the one hand, consistency and maintainability for the content, while on the other hand, they also provide a wide set of multi-modal, interactive input/output interfaces. In a modern changing and complex assembly production environment, it is also important that the WIS is connected to the execution control and supervisory systems, which provide information regarding the shop floor status, thus tailoring the delivered information to the actual context.

The approach of the paper follows the workflow shown in Fig. 3, which explains how WIS is involved in the generation and the context aware delivery of instructions. The aim of automated instruction generation is to create animated 3D and textual content. An assembly process engineer can use this content as a basis for finalizing the instructions (by tailoring or extending them with additional content, e.g., videos, pictures, etc.). However, during instruction generation, it has to be taken into consideration that workers may have different skills, language proficiency and content delivery devices at hand. These properties define the part of the worker context which needs to be addressed during instruction generation. After content generation, instructions are delivered by the WIS according to the complete context, which is defined by the following aspects:

- Process context: the task to be executed, which triggers the instruction delivery, coming from execution tracking.
- Worker activity context: actions performed by the workers which form commands toward the execution tracking system. The commands (e.g., acknowledgement, failure) are interpreted by the WIS and sent to the execution tracking system.
- Worker location context: actual position of the worker in the shop floor, which identifies the worker in the workcell, thus enables delivering location-aware instructions (e.g., increasing font size or volume when the worker is further away from the devices).
- Worker properties context: skills, preferences of the worker. This information, stored in the worker database, has to be taken into account during instruction generation and delivery in order to ensure that the displayed instructions match the requirements of the given worker.

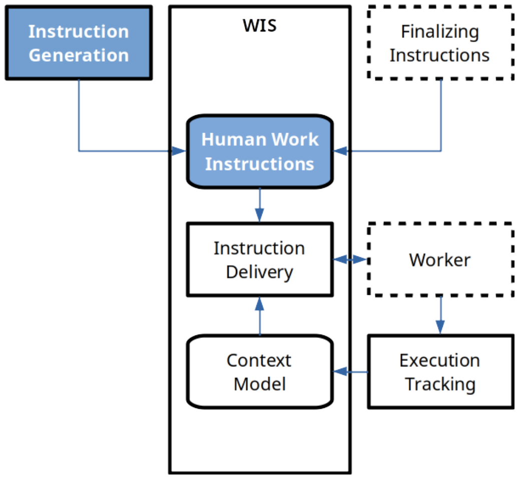


Figure 3: Overview of WIS's role in generation and context-aware delivery of instructions. The connection to the workcell configuration is highlighted in blue.

For the automatic generation of work instructions the features serve as basic templates of instructions. Each individual feature-based task defines the micro-world of its execution. In the presented approach, instructions are skill level-dependent: more skilled workers only require the core instructions (highlighted in bold in the template below). The following hierarchical template structure is used for generating instructions from the feature-based plan representation:

1. simple_feature_task(feature, base_part, moved_part, skill)

1.1. approach(location(moved_part)) - Approaching location

1.2. pick_up(moved_part) - Picking up part

1.3. **core_task(base_part, moved_part, type(feature))** - Performing the core corresponding to the feature type

- * insert(base_part, moved_part) - Inserting moved part into base part
- * place(base_part, moved_part) - Placing moved part to base part
- * screw(base_part, moved_part) - Inserting and tightening screw (moved part) into base part

2. changeover_task(old_equipment, new_equipment), [part], skill)

2.1. release(old_equipment) - Releasing installed equipment (tool or fixture)
2.2. pick_up(old_equipment)
2.3. approach(pick_up_location(old_equipment))
2.4. put_away(old_equipment)
2.5. approach(pick_up_location(new_equipment))
2.6. pick_up(new_equipment)
2.7. **install(new_equipment, [part])** - Install tool / Grasp part in fixture

3. composite_feature_task({simple_feature}, ordering, skill)
   - ordering - Specifying sequence for features inside composite features
     * *parallel*
     * *sequential*
     * *arbitrary*

3.1. **composite_header(simple_feature_task(), ordering)** - Performing multiple tasks in composite feature based on the first task according to ordering
3.2. repeat(simple_feature, ordering) - Performing a list of simple features according to ordering

For the above hierarchical approach to work, the following helper functions are required:

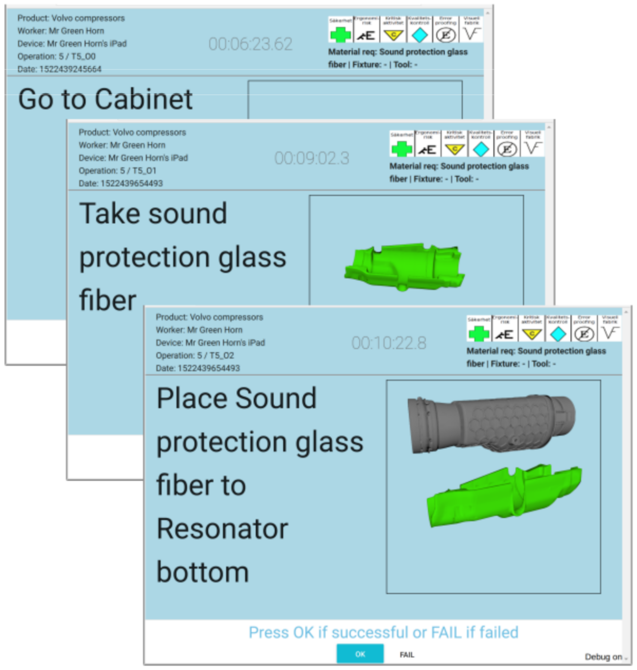
- subassembly(part) - Returns the subassembly to which the part belongs
- pick_up_location(object) - Returns the pick-up location of the object
- put_away_location(object) - Returns the put away location of the object
- type(simple_feateure) - Returns the type of the simple feature

Generating Textual Instructions

Textual instructions traditionally serve as the elementary method for delivering information to workers. The paper-based and the newer digital approaches are both limited by the human effort required to keep them up-to date, which opens up potential for automated or semi-automated textual instruction generation.

Textual instructions are generated for task execution by using the following instruction templates and the data from the tasks:

- approach(location): "Go to *_location*"
- pick_up(object): "Take *object* [in subassembly *subassembly(object)*]"
- insert(base_part, moved_part): "Insert part *moved_part* [in subassembly *subassembly(move_part)*] into *base_part* [in subassembly *subassembly(base_part)*]"
- place(base_part, moved_part): "Place part *moved_part* [in subassembly *subassembly(move_part)*] to *base_part* [in subassembly *subassembly(base_part)*]"
- screw(base_part, moved_part): "Insert screw *moved_part* into *base_part* [in subassembly *subassembly(base_part)*] and tighten it"


Product: Volvo compressors
Worker: Mr Green Horn
Device: Mr Green Horn's iPad
Operation: 5 / T5_O0
Date: 1522439245664
00:06:23.62
Material req: Sound protection glass fiber | Fixture: - | Tool: -
Go to Cabinet

Product: Volvo compressors
Worker: Mr Green Horn
Device: Mr Green Horn's iPad
Operation: 5 / T5_O1
Date: 1522439654493
00:09:02.3
Material req: Sound protection glass fiber | Fixture: - | Tool: -
Take sound protection glass fiber

Product: Volvo compressors
Worker: Mr Green Horn
Device: Mr Green Horn's iPad
Operation: 5 / T5_O2
Date: 1522439654493
00:10:22.8
Material req: Sound protection glass fiber | Fixture: - | Tool: -
Place Sound protection glass fiber to Resonator bottom
Press OK if successful or FAIL if failed

Figure 4: An example of a task delivered to a less skilled worker in more details.

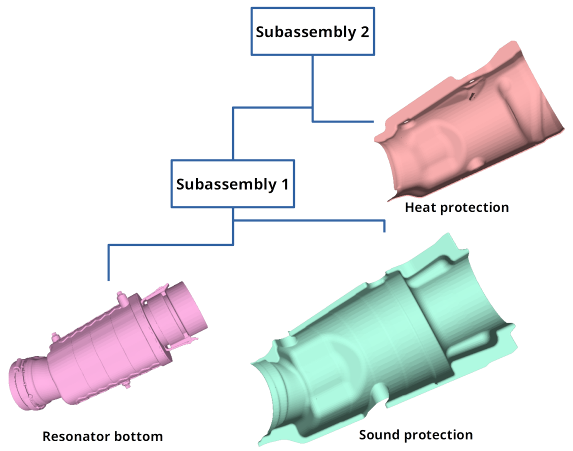
- composite_header(simple_feature_task, ordering): "Execute on all items in the *ordering* order."
- release(object): "Release *object*"
- grasp(fixture, part): "Grasp *part* [in subassembly *subassembly(part)*] in fixture *fixture*"

For workers with advanced skill levels, only the core assembly instructions are delivered using repetition and ordering flags for composite instructions, while for workers in training every step is displayed (e.g., approach, pick-up, core instruction). This is in line with industrial practice, where the supervision of task execution is done on a lower level for workers in training, hence it is required to display and have every step acknowledged. This approach also allows using a single workflow for generating content for both beginner and experienced workers.

Generating Visual Instructions

Micro-level planning enables using the geometrical models in generating animated instructions as well. In order to implement this, X3D format was chosen for displaying 3D instruction content.

X3D is an extension of the VRML format and is part of the HTML5 standard, which means content can be displayed and manipulated through web-browsers. It also supports linking geometries to reference frames and thus enables building up kinematic chains. Using this approach the assembly tree and its components are represented in a multiple level reference frame hierarchy. Additionally, rearranging the nodes of the model is done through the HTML's Document Object Model (DOM) (e.g., attaching the model of



```
<X3D>
 <Scene>
  <Transform DEF="Subassembly␣2">
   <Transform DEF="Subassembly␣1">
    <Transform DEF="Heat␣protection">
     <Shape><IndexedTriangleSet ... ></Shape>
    </Transform>
    <Transform DEF="Resonator␣bottom">
     <Shape><IndexedTriangleSet ... ></Shape>
    </Transform>
   </Transform>
   <Transform DEF="Sound␣protection">
    <Shape><IndexedTriangleSet ... ></Shape>
   </Transform>
  </Transform>
 </Scene>
</X3D>
```

Table 1: A sample assembly tree with three parts and the structure of the corresponding X3D representation.

the tool to the moved object).

The geometric models applied in planning are triangle mesh models, which offer generic and open access representation. These models can be easily translated into the *(indexed) triangle set* representation of the X3D format. Moreover, as the required resolution for displaying instruction is lower than that of planning, in order to enhance the rendering speed, the 3D models are resampled by applying edge decimation algorithms over the triangle meshes (e.g., quadratic edge collapse (Garland and Heckbert 1997), see Fig. 5).

Each geometry in the model (parts, tools, fixtures) are translated into X3D format in such a way that the *shape* node containing the *indexed face set* representation of the geometry is appended to a *transform* node. The *transform* node allows the application of geometric transformation to its children *shapes* by exposing its *translation* and *orientation* fields. Similarly to parts, each node in the assembly tree (i.e., subassembly) is represented by its own *transform* node and therefore contains all the previously assembled parts of its branch (see Table 1).

Building up the X3D scene in a structure symmetrical to the assembly tree enables setting attributes of already assembled parts in either one field at the top of the hierarchy, or by searching recursively via the DOM structure of the X3D-scene (e.g., by using *jquery*). This applies to transformations (position and orientation, exposed by the *transform* node; note that planning assumes that parts once assembled remain in this state) and to display options such as switching rendering and changing color. Hence, the scene (the displayed objects and their position and orientation) can be set to suit a given state defined by an assembly step. Also, the models of tools and fixtures are attached to the corresponding parts or subassemblies dynamically, according to the displayed step. Each assembly step is described by the following data:

- ID of the assembly step
- IDs of the moved parts/subassemblies
- IDs of the base parts/subassemblies
- Transformation of the near position
- Nodes of the path
- ID of the tool
- Tool Center-Point Frame (TCPF) transformation of the tool
- ID of the fixture
- Transformation of the fixture

For displaying 3D instructions two cases are distinguished: (1) showing a static view of components in assembled or disassembled (near position) state; (2) animation of movement provided by path planning. In both cases, it might be required to show only the base and moved components, therefore turning off rendering for all the nodes which are not in the branch of the assembly tree starting from the subassembly node of the actually displayed assembly step. The implemented visual instructions are the following:

- approach(location): static view of location
- pick_up(object): static view of object
- insert(base_part, moved_part): animation of local and approach motion
- place(base_part, moved_part): animation of local and approach motion
- screw(base_part, moved_part): animation of local and approach motion
- composite_header(simple_feature, ordering): visualization according to the underlying simple features and ordering flag
- release(object): static view of object
- grasp(fixture, part): static view of fixture and part

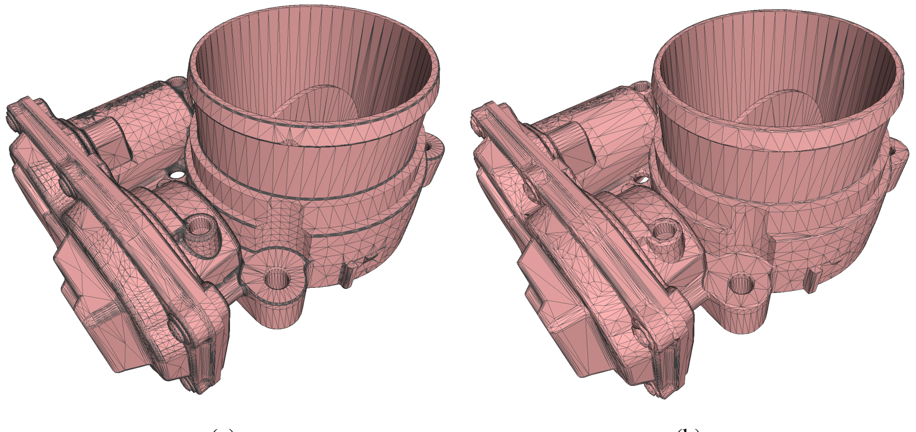

(a) (b)

Figure 5: Two triangle mesh models, representing the same object before (a) and after (b) resampling. By applying a 0.9 reduction the number of vertices were reduced from ˜90k to ˜10k, which has a significant impact on the rendering speed.

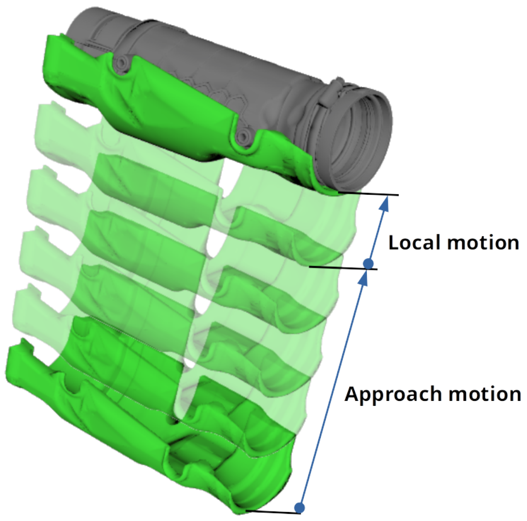


Figure 6: Illustration of X3D animation of two components being assembled, the moved component is shown in green.

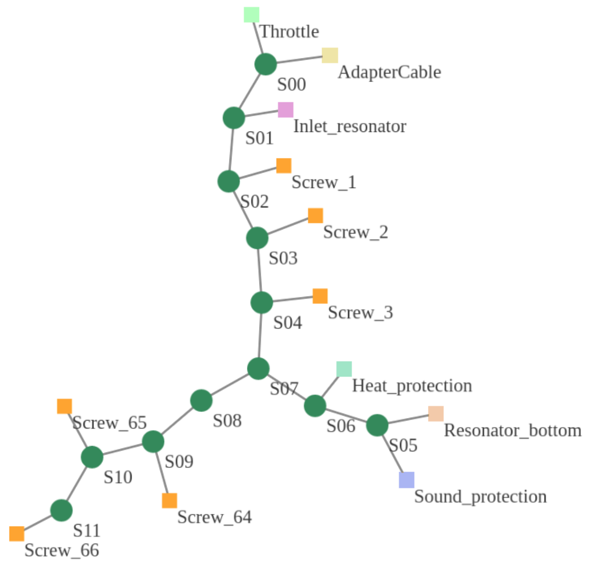


Figure 7: Assembly process plan for the case study, showing the assembly steps from S00 to S11. Parts in composite features (i.e., screws) are highlighted in orange.

As there is no animation in case (1), the assembled and the disassembled state can be displayed by simply applying the transformation of the near position to the *transform* node of the moved component. The transformation applies to all the children nodes (i.e., parts or subassemblies). To create animated movements along the nodes of the path, the *Interpolator* X3D node is utilized, which implements linear interpolation between each node (see Fig. 6). The duration of the interpolation is calculated assuming a constant speed along the path.

Case Study

The proposed approach is demonstrated in a case study from the automotive industry. The assembly, namely the *inlet bypass* is part of a so-called *supercharger* component composed of 29 parts. The *inlet bypass* is built up by 12 parts: 6 main components and 6 screws (see Fig. 8). There are five simple features and two composite features, either of them involving joining a set of parts by 3-3 screws, and one auxiliary task. Solving the CAPP model of the problem resulted in the assembly sequence shown in Fig. 7, which minimizes the time of tool and fixture changeovers.

The demonstration is part of a research project aiming to develop a WIS for supporting multi-modal and context-

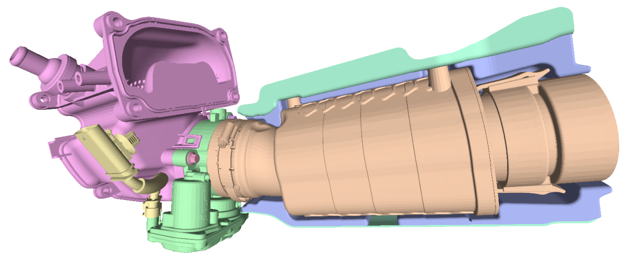


Figure 8: The assembly of the case study contains 12 components (6 screws). The coloring of the parts is similar to that in the assembly process plan (Fig. 7), except for the screws in the composite features.

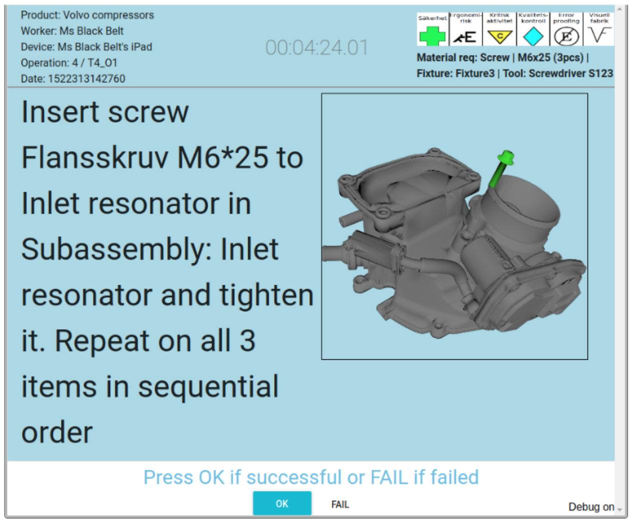
aware instruction delivery in flexible workcells. The developed system is connected to a workcell-level controller, which controls and tracks task execution and triggers instruction delivery to multiple devices. The WIS has a frontend-backend architecture, where the content database of the backend is populated with the automatically generated instructions based on the CAPP solution. The frontend is responsible for delivering the instructions to the devices of the worker (e.g., smartphone, tablet, screen, headphones). The visual instructions are delivered through a responsive HTML5 web-page, which also utilizes a text-to-speech library for the audio delivery of the textual instructions.

In the case study, instructions were generated for workers with two different skill levels. One is the worker with advanced skills, who receives only the core instructions, the other is the worker under training, who receives more detailed instructions. Figs. 4 and 9 show screenshots of the WIS frontend. In addition to displaying the generated instructions, the WIS frontend can also be configured to display safety or quality symbols (see the upper right corner of Figs. 4 and 9). The research project is now in its closing phase, where the focus is on the development of demonstrator case studies and the evaluation of the perceived work experience with using the generated content and the multimodal content delivery system.

Conclusions

This paper presented a novel approach to the automated generation and the presentation of context-sensitive work instructions for human workers in mechanical assembly. The methods have been implemented and integrated into a module of a comprehensive assembly workcell configuration toolbox, to visualize the assembly plans generated by a mixed-initiative, optimizing process planner. The generated work instructions contain both textual instructions, delivered either visually on a screen or as an audio stream using text-to-speech tools, as well as X3D animations. The main advantage of the approach is the ability to efficiently generate and present customized, context-sensitive instructions that take into account the workers' skill levels, language, devices and presentation preferences.

In order to proceed from the current, prototype-level implementation to an industrial deployment, future work must



Product: Volvo compressors
Worker: Ms Black Belt
Device: Ms Black Belt's iPad
Operation: 4 / T4_01
Date: 1522313142760

00:04:24.01

Material req: Screw | M6x25 (3pcs) |
Fixture: Fixture3 | Tool: Screwdriver S123

Insert screw Flansskruv M6*25 to Inlet resonator in Subassembly: Inlet resonator and tighten it. Repeat on all 3 items in sequential order

Press OK if successful or FAIL if failed

Figure 9: An example of a composite task delivered to an expert worker using the repetition and ordering flags.

focus on extending the instructions to satisfy all relevant industrial standards, such as placing identifiers and safety symbols on the instruction screens, preferably in an automatically generated way. In a wider context, the further development of various planning functions in the overall assembly workcell configuration toolbox holds numerous research challenges. Above all, a stronger support and integration is required for the generation of the feature-based assembly model from legacy design representations.

Acknowledgements

This research has been supported by the GINOP-2.3.2-15-2016-00002 grant on an “Industry 4.0 research and innovation center of excellence” and by the EU H2020 Grant SYMBIO-TIC No. 637107.

References

Bader, S., and Aehnelt, M. 2014. Tracking assembly processes and providing assistance in smart factories. 161–168. SCITEPRESS—Science and and Technology Publications.

Bannat, A.; Wallhoff, F.; Rigoll, G.; Friesdorf, F.; Bubb, H.; Stork, S.; Müller, H. J.; Schubö, A.; Wiesbeck, M.; and Zäh, M. F. 2008. Towards optimal worker assistance: A framework for adaptive selection and presentation of assembly instructions. In *Proceedings of the 1st international workshop on cognition for technical systems, Cotesys.*

Claeys, A.; Hoedt, S.; Landeghem, H. V.; and Cottyn, J. 2016. Generic model for managing context-aware assembly instructions. *IFAC-PapersOnLine* 49(12):1181–1186.

Fiorentino, M.; Uva, A. E.; Gattullo, M.; Debernardis, S.; and Monno, G. 2014. Augmented reality on large screen for interactive maintenance instructions. *Computers in Industry* 65(2):270–278.

Garland, M., and Heckbert, P. S. 1997. Surface simplification using quadric error metrics. In *Proceedings of the*

24th annual conference on Computer graphics and interactive techniques, 209–216. ACM Press.

Ghandi, S., and Masehian, E. 2015. Review and taxonomies of assembly and disassembly path planning problems and approaches. *Computer-Aided Design* 67-68:58–86.

Hu, S.; Ko, J.; Weyand, L.; ElMaraghy, H.; Lien, T.; Koren, Y.; Bley, H.; Chryssolouris, G.; Nasr, N.; and Shpitalni, M. 2011. Assembly system design and operations for product variety. *CIRP Annals* 60(2):715–733.

Kaipa, K.; Morato, C.; Zhao, B.; and Gupta, S. K. 2012. Instruction Generation for Assembly Operations Performed by Humans. In *32nd Computers and Information in Engineering Conference, Parts A and B*, volume 2, 1121–1130. ASME.

Kardos, C.; Kovács, A.; and Váncza, J. 2016. Towards feature-based human-robot assembly process planning. *Procedia CIRP* 57:516–521.

Kardos, C.; Kovács, A.; and Váncza, J. 2017. Decomposition approach to optimal feature-based assembly planning. *CIRP Annals—Manufacturing Technology* 66(1):417–420.

Leu, M. C.; ElMaraghy, H. A.; Nee, A. Y.; Ong, S. K.; Lanzetta, M.; Putz, M.; Zhu, W.; and Bernard, A. 2013. CAD model based virtual assembly simulation, planning and training. *CIRP Annals—Manufacturing Technology* 62(2):799–822.

Lušić, M.; Schmutzer Braz, K.; Wittmann, S.; Fischer, C.; Hornfeck, R.; and Franke, J. 2014. Worker information systems including dynamic visualisation: A perspective for minimising the conflict of objectives between a resource-efficient use of inspection equipment and the cognitive load of the worker. *Advanced Materials Research* 1018:23–30.

Lušić, M.; Fischer, C.; Bönig, J.; Hornfeck, R.; and Franke, J. 2016. Worker information systems: State of the art and guideline for selection under consideration of company specific boundary conditions. *Procedia CIRP* 41:1113–1118.

Michalos, G.; Makris, S.; Spiliotopoulos, J.; Misios, I.; Tsarouchi, P.; and Chryssolouris, G. 2014. ROBO-PARTNER: Seamless human–robot cooperation for intelligent, flexible and safe operations in the assembly factories of the future. *Procedia CIRP* 23:71–76.

Pan, J.; Chitta, S.; and Manocha, D. 2012. FCL: A general purpose library for collision and proximity queries. In *IEEE International Conference on Robotics and Automation*, 3859–3866.

Wang, L.; Keshavarzmanesh, S.; and Feng, H.-Y. 2011. A function block based approach for increasing adaptability of assembly planning and control. *Int J of Production Research* 49(16):4903–4924.

MA-RADAR – A Mixed-Reality Interface for Collaborative Decision Making

Sailik Sengupta[*] and **Tathagata Chakraborti**[*] and **Subbarao Kambhampati**

School of Computing, Informatics, and Decision Systems Engineering
Arizona State University

{ sailiks, tchakra2, rao } @ asu.edu

Abstract

There has been a lot of recent interest in the planning community towards adapting automated planning techniques for the role of decision support for human decision makers in the loop. A unique challenge in such settings is the presence of multiple humans collaborating during the planning process which not only requires algorithmic advances to handle issues such as diverging mental models and the establishment of common ground, but also the development of user interfaces that can facilitate the distributed decision making process among the human planners. We posit that recent advances in augmented reality technology is uniquely positioned to serve this need. For example, a mixed-reality workspace can be ideal for curating information towards the particular needs (e.g. explanations) of the individual decision makers. In this paper, we report on ongoing work along these directions and showcase MA-RADAR, the multi-agent version of the decision support system RADAR (Sengupta et al. 2017).

In (Sengupta et al. 2017), we explored the evolving roles of an automated planner in the scope of decision support for a single human in the loop. Specifically, we outlined how well-established principles of *naturalistic decision-making* and the *automation hierarchy* studied in existing literature on human-computer interaction (Parasuraman, Sheridan, and Wickens 2000; Klein 2008) can be adopted for the design of automated decision support using planners as well. In this regard, we demonstrated how the traditional role of an automated planner changes from one of plan generation to more nuanced roles of plan validation, recognition, recommendation, critique, explanations, and so on. However, most of these techniques, as well as the GUI itself, were specifically designed to deal with a single human decision maker in the loop. As we illustrate in the paper, these become ineffective in a distributed setting.

What if there are *multiple humans* in the loop?

A common feature of most collaborative planning settings is the presence of multiple human decision makers who are actively involved in the construction of the plan on a shared graphical user interface (GUI) in “control room” styled environments (Chakraborti et al. 2017b; Karafantis 2013; Murphy 2015). For the design of decision support technologies, this raises several unique challenges such as (1) dealing with diverse points of view, preferences, and goals; (2) diverging beliefs and mental models; (3) resolution of competing truths and establishment of common ground; and so on. Some of these issues have been highlighted recently in (Kim and Shah 2017). From the perspective of the GUI itself, the presence of multiple decision makers poses new challenges on how information is presented to the end users, not only in the way it is displayed, but also the approach to generate that information which drives the decision support infrastructure in the back-end. Recent advances in augmented-reality technologies in opening up newer channels of communications with AI agents (Williams et al. 2018) can begin to address some of these challenges.

What can augmented reality bring to the table?

We argue that augmented reality (AR) brings in capabilities that are uniquely suited for this purpose. This is because AR can, in effect, provide different versions of the same interface to the commanders based on their specific needs, while still preserving the convenience and efficiency of collaboration across a shared GUI. In this work, we thus will build on our previous decision support system – c.f. RADAR (Sengupta et al. 2017) – and highlight challenges, especially as it relates to the design of the interface for the decision support system, when the collaborative decision making setting is extended to deal with multiple human planners simultaneously. We will, in particular, show how –

- Augmented reality provides an effective medium of augmenting the shared GUI with private information (as studied in planning literature (Brafman and Domshlak 2008)) – thus the same plan will appear differently on the shared GUI than in the mixed-reality view where the private actions will be coupled with the public plan; and
- Augmented reality can reduce irrelevant information on the screen by porting them into the mixed-reality view. Such situations can occur, for example, when one user asks for an explanation, which the others may not require and thus should not appear on the shared GUI and potentially cause cognitive overload.

Finally, we will end with a discussion on the current work in progress and considerations of the trade-offs in AR versus distributed graphical interfaces. We note, as far as the *vitamin versus aspirin* question is concerned, AR firmly

[*]Authors marked with [*] contributed equally.

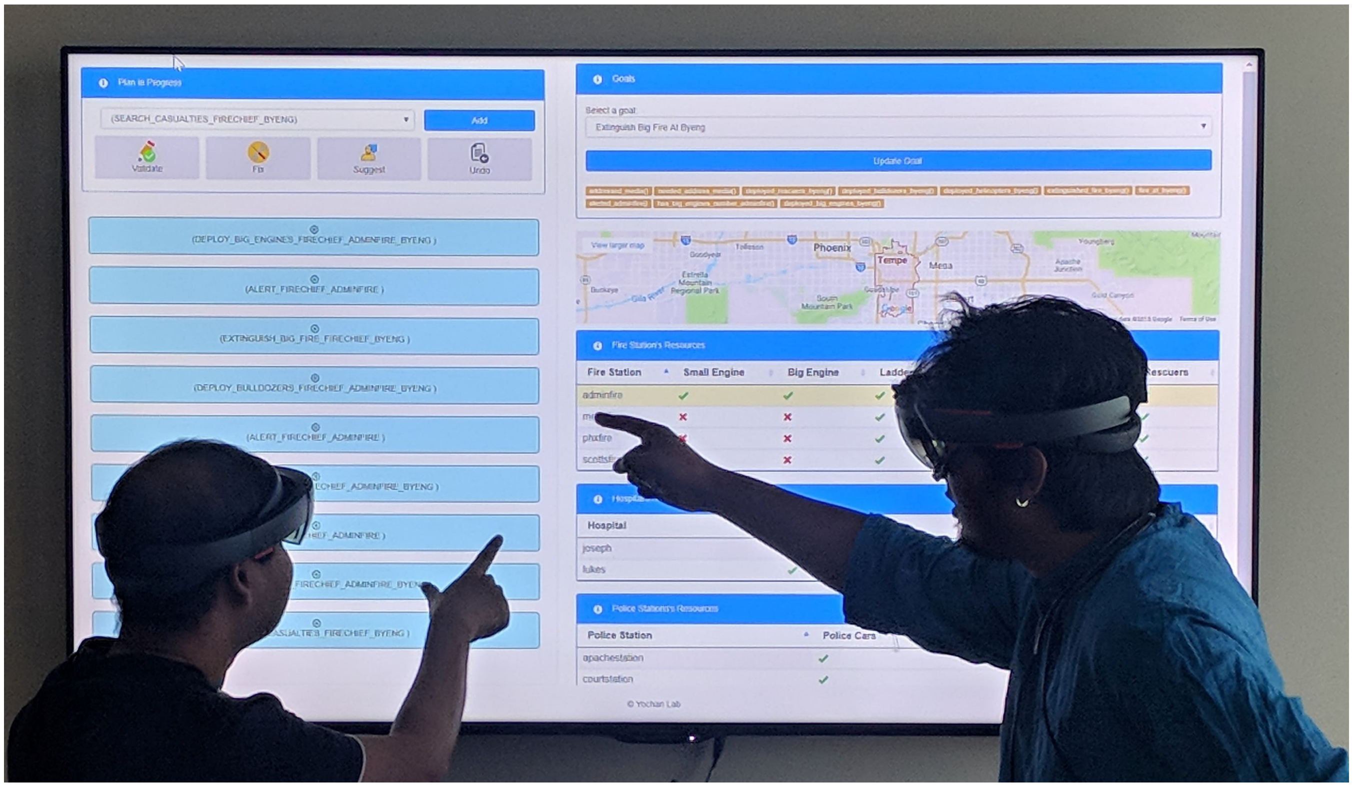

Plan in Progress
(SEARCH_CASUALTIES_FIRECHIEF_BYENG)
Add
Validate
Fix
Suggest
Undo
(DEPLOY_BIG_ENGINES_FIRECHIEF_ADMINFIRE_BYENG)
(ALERT_FIRECHIEF_ADMINFIRE)
(EXTINGUISH_BIG_FIRE_FIRECHIEF_BYENG)
(DEPLOY_BULLDOZERS_FIRECHIEF_ADMINFIRE_BYENG)
(ALERT_FIRECHIEF_ADMINFIRE)
Goals
Select a goal
Extinguish Big Fire At Byeng
Update Goal
Phoenix
Tempe
Mesa
Fire Station's Resources
Fire Station
Small Engine
Big Engine
Rescuers
adminfire
phxfire
Hospital
joseph
lukes
Police Station's Resources
Police Station
Police Cars
apachestation
courtstation
© Yochan Lab

Figure 1: Multiple commanders involved in the collaborative decision making process on the MA-RADAR interface. The shared interface (GUI) provides an overview of the public plan and resources, constraints, etc. pertaining to the planning problem. The mixed-reality view for each commander augments private information (such as private action in the plan) or personalized explanations of the plan for the commander. Refer to Figures 2 and 3 for the augmented view for each of these use cases.

lies with the former group – after all, the humans could be equipped with separate personal screens on top of a shared GUI. However, we argue, and hopefully this is apparent in the demonstrations as well, that AR provides an attractive solution towards providing personalized planning interfaces to the human decision makers while still leveraging the paradigm of a shared collaborative interface of a control room accepted as the de-facto standard in these settings.

MA-RADAR

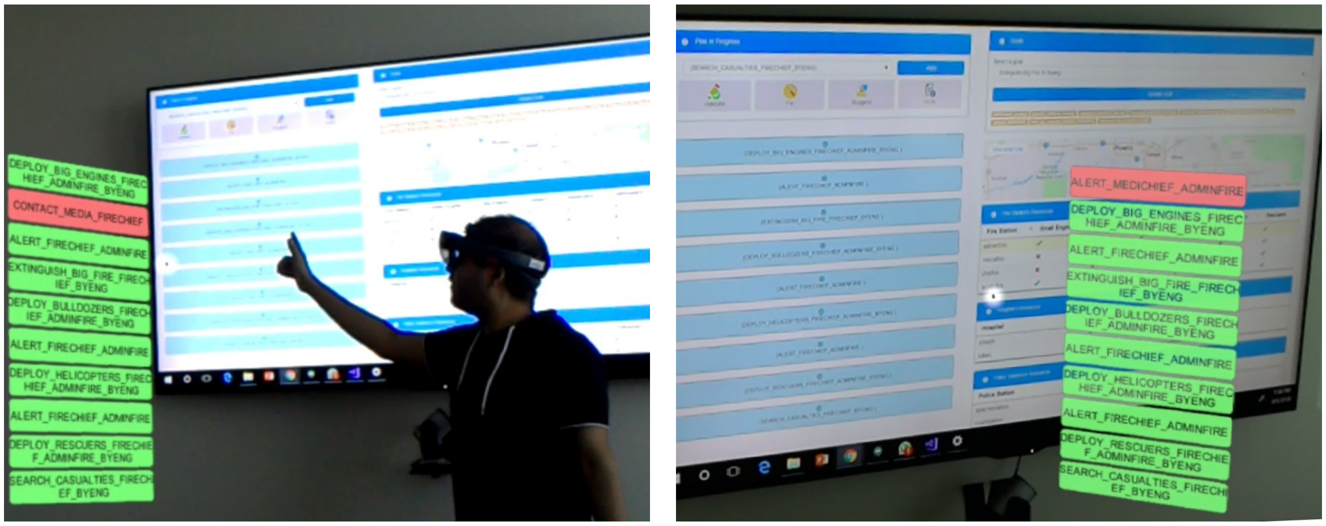
In the following, we will briefly introduce the fire fighting domain (Sengupta et al. 2017) which we will use to illustrate the UI challenges addressed in the paper.

The Fire-Fighting Domain The fire fighting domain involves extinguishing fire at a particular location (Tempe, in our case). It requires two commanders (henceforth referred to as Comm-I and Comm-II) to come up with a plan or course of action (CoA) which involves coordination with the police, medical and transport authorities. Each commander might have a personalized model of this domain, which (1) may have certain actions that are *private* to them, i.e. unknown to the other commanders; and (2) incorrect ideas about the actual domain, for example, an incorrect action definition (according to the model of the decision support agent). A detailed description of the domain used by MA-RADAR is available in (Sengupta et al. 2017). We assume that these personalized models (of the two experts) are available to MA-RADAR, which helps it to distinguish between private and public actions (Brafman and Domshlak 2008). While MA-RADAR uses a centralized model to help in validating plans and generating action or plan suggestions, the response to the users needs to be carefully curated since showing one user's private data to another user is problematic. Furthermore, explanations are inherently user specific (as are information that is being consumed by private actions only) and should not clutter a shared GUI.

Privacy Preserving Planning

In the first demonstration, we will tackle the issue of private information in the individual models of the commanders.

Background In (Brafman and Domshlak 2008) authors explored multi-agent planning scenarios where each agent has a different domain model with individual actions that can have private preconditions and effects which are not accessible to other agents. Planning in such scenarios becomes more complex because state-space search techniques have to ensure that private state variables of an agent are not exposed to other agents (Brafman 2015). As mentioned before, the interface in MA-RADAR follows this notion of private and public predicates/actions and communi-


DEPLOY_BIG_ENGINES_FIREC HIEF_ADMINFIRE_BYENG
CONTACT_MEDIA_FIRECHIEF
ALERT_FIRECHIEF_ADMINFIRE
EXTINGUISH_BIG_FIRE_FIRECH IEF_BYENG
DEPLOY_BULLDOZERS_FIRECH IEF_ADMINFIRE_BYENG
ALERT_FIRECHIEF_ADMINFIRE
DEPLOY_HELICOPTERS_FIREC HIEF_ADMINFIRE_BYENG
ALERT_FIRECHIEF_ADMINFIRE
DEPLOY_RESCUERS_FIRECHIE F_ADMINFIRE_BYENG
SEARCH_CASUALTIES_FIRECHI EF_BYENG
ALERT_MEDICHIEF_ADMINFIRE
DEPLOY_BIG_ENGINES_FIREC HIEF_ADMINFIRE_BYENG
ALERT_FIRECHIEF_ADMINFIRE
EXTINGUISH_BIG_FIRE_FIRECH IEF_BYENG
DEPLOY_BULLDOZERS_FIRECH IEF_ADMINFIRE_BYENG
ALERT_FIRECHIEF_ADMINFIRE
DEPLOY_HELICOPTERS_FIREC HIEF_ADMINFIRE_BYENG
ALERT_FIRECHIEF_ADMINFIRE
DEPLOY_RESCUERS_FIRECHIE F_ADMINFIRE_BYENG
SEARCH_CASUALTIES_FIRECHI EF_BYENG

(a) Comm-I, who is responsible for communication with the media, has a private action to contact media as visible only in his POV.

(b) Comm-II, who is in charge of communicating with the medical units, has a private action to alert the medical chief in the area.

Figure 2: Mixed-reality capture illustrating how the public plan in the shared GUI can be overlayed with information on private actions (private actions are in red; public actions are in green) (Brafman and Domshlak 2008) of individual decision makers thereby still allowing the use of a shared collaborative workspace.

cates information (e.g. explanations and plan suggestions) to the user without revealing private data of another human in the team. Note that, from the point of view of decision support, the agent itself is not following planning algorithms as outlined in (Brafman and Domshlak 2008; Brafman 2015) since the human planners are in charge of the planning process and they, of course, are not maintaining separate priority queues in their heads. However, it might be interesting to explore how the distributed planning paradigm among humans in the presence of private information can be modeled from the perspective of decision support.

Demonstration For the purpose of our demonstration (shown in Figure 2), we assume that apart from the main task of extinguishing the fire, each of the commanders have specific tasks they need to achieve. Furthermore, only the commander (in charge of a specific task) and `MA-RADAR` have the knowledge of these private tasks.

In our scenario, while Comm-I is in charge of handling the communication with the media, which is an important aspect in the case of disaster response scenario, Comm-II needs to take care of all communication and deployment of medical help for rescued victims. The private actions of the two commanders follow. To reduce clutter, we only provide the action names as opposed to the full action definition (which also have the private predicates).

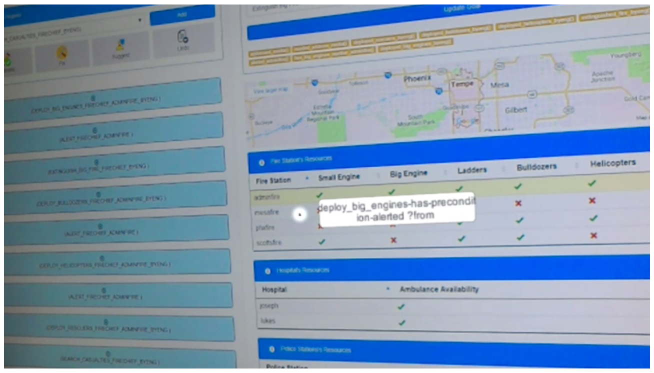
```
Comm-I
CONTACT_MEDIA
ADDRESS_MEDIA

Comm-II
ALERT_MEDICHIE
ATTEND_CASUALTIES
ISSUE_LOCAL_ALERT
SET_UP_HELPLINE
```

When the commanders ask `MA-RADAR` to suggest missing actions or complete the plan in order to achieve the goal of extinguishing the (big) fire, it needs to communicate most of the private actions mentioned here, but only to the specific commander in charge of the private task. Showing these on the common user interface would result in (1) confusion as each commander is oblivious to the private actions of the other commander and (2) loss of privacy which might be important in complex decision making scenarios with multiple commanders (e.g. army, navy, etc.).

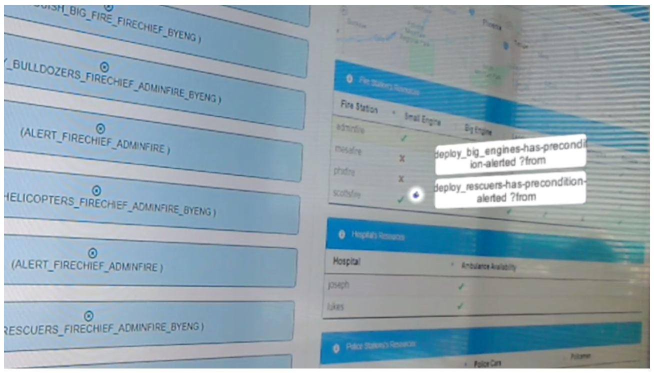
Thus, in such scenarios, `MA-RADAR` tries to display these private actions in the augmented view of each commander (highlighted in red in Figure 2). Since each action in the plan occupies a substantial amount of space in the 3D-view, we show only 10 actions at any point in time. This ensures that the commander does not have to stare away from the common interface, which can lead to loss of situational awareness as other commanders might make changes to common elements in that time (e.g. by updating resources, rearranging or removing public actions, etc.).

Multi-Model Explanations

The second demonstration looks at *plan explanations for model reconciliation* introduced in (Chakraborti et al. 2017a) – the aim of explanations of this form is to provide updates to the user's possibly faulty understanding of the planning problem to make sure that the optimal plans in the planner's model are also optimal in the human's. Thus the process of model reconciliation is crucial in maintaining that the decision support agent is on the same page as the human in the loop and thus the establishment of common ground.



(a) Comm-I, who is unaware of the procedure that a fire-chief needs to be alerted first before deploying the fire engines, is provided this explanation to justify the suggested (public part of the) plan.


deploy_big_engines-has-precondition-alerted ?from
deploy_rescuers-has-precondition-alerted ?from

(b) Comm-II, unaware that a fire-chief needs to be alerted before deploying any kind of resources (fire engines or rescuers) from a fire station, is provided both of these model updates as explanations.

Figure 3: Mixed-reality capture illustrating how the multi-model explanation generation algorithm (Sreedharan, Chakraborti, and Kambhampati 2018) can be used to provide targeted explanations to each commander based on their models without inundating the other commanders with superfluous or unsolicited information.

Background The above model reconciliation process is only feasible if inconsistencies of the planner's model with the user's mental model are known precisely, or in general, if there is a single model that needs to be reconciled. Instead, in a team decision making setting, the decision support agent may end up having to explain its decisions with respect to a *set of models* one for each human in the loop. In this situation, MA−RADAR can look to compute explanations for each possible configuration. However, computing separate explanations (Chakraborti et al. 2017a) for each agent can result in situations where the explanations computed for individual models independently are not consistent across all the possible target domains. In the case of multiple teammates being explained to, this may cause confusion and loss of trust, especially in teaming with humans who are known to rely on shared mental models. Instead, in (Sreedharan, Chakraborti, and Kambhampati 2018) we proposed an explanation generation process such that a single model update that makes the given plan optimal (and hence explained) in all the updated domains (or in all possible domains).

In order to deal with multiple humans in the loop, we have thus shifted towards (Sreedharan, Chakraborti, and Kambhampati 2018) instead of (Chakraborti et al. 2017a) as originally demonstrated in (Sengupta et al. 2017). However, from the point of view of the interface, there still remains the matter of filtering out superfluous information (due to the single explanation or model update being computed that suffice for all the models) as they are being presented to the individual users. We will illustrate this next.

Demonstration For our demonstration (shown in Figure 3), we assume that the two commanders have different understanding about the domain used by MA−RADAR. We further assume that this knowledge about the actual domain is (1) different for the different commanders, which is often the case in real-world scenarios (as there may be many ways of being incorrect about the correct procedure) and (2) these explanations are, for the purpose of this example, limited to updates about public actions. In scenarios where explanations are about private predicates or private actions of a commander, MA−RADAR , with the models of both the commanders, filters out these (private explanations) when generating explanations for the other commander.

In order to highlight the domain differences, we will first show the part of the actual model (that MA−RADAR has) about which the commanders have incorrect idea.

```
(:action deploy_big_engines
  :parameters (?a - fire ?from -
    firestation ?to - pois)
  :precondition (and
    (alerted ?from)
    (has_big_engines_number ?from)
  )
  :effect (and
    (not (alerted ?from))
    ...
  )
)
(:action deploy_rescuers
  :parameters (?a - fire ?from -
    firestation ?to - pois)
  :precondition (and
    (alerted ?from)
    (has_rescuers_number ?from)
  )
  :effect (and
    ...
  )
)
```

We now show the model that Comm-I has, where the precondition for alerting the authority at a fire-station is missing as a precondition for deploying (big) fire engines –

```
(:action deploy_big_engines
    :parameters (?a - fire ?from -
        firestation ?to - pois)
    :precondition (and
        (has_big_engines_number ?from)
        )
    :effect (and
      (not (alerted ?from))
      ...
    )
)
(:action deploy_rescuers
    :parameters (?a - fire ?from -
        firestation ?to - pois)
    :precondition  (and
      (alerted ?from)
      (has_rescuers_number ?from)
    )
    :effect (and
      ...
    )
)
```

For Comm-II, who is completely unaware that fire-stations need to be alerted in order to deploy fire engines or rescuers, the domain model looks as follows –

```
(:action deploy_big_engines
    :parameters (?a - fire ?from -
        firestation ?to - pois)
    :precondition (and
        (has_big_engines_number ?from)
        )
    :effect (and
      (not (alerted ?from))
      ...
    )
)
(:action deploy_rescuers
    :parameters (?a - fire ?from -
        firestation ?to - pois)
    :precondition (and
        (has_rescuers_number ?from)
        )
    :effect (and
      ...
    )
)
```

When the commanders ask MA-RADAR to suggest a plan (or complete a plan) in order to achieve the goal of extinguishing big fire, it will suggest a plan that has both the actions of deploying big engines and rescuers. Since both of these actions need to alert the authority at the fire station, there will be two alert_firechief actions which makes the alerted_firechief proposition (which is a precondition of these two actions in the original domain) true.

In this situation, although both the commanders might be surprised at the suggested plan and ask for explanations, Comm-I just needs to be told about the missing precondition of the deploy_big_fire_engine action, whereas, Comm-II, in addition to that explanation, also needs to be told about the missing precondition of the action deploy_rescuers. The augmented reality workspace helps us to provide personalized explanations to both the commanders (see Figure 3).

Work in Progress

Currently, we are working on making the mixed-reality display more interactive and porting more of the utilities in the shared GUI into it. This, of course, raises interesting challenges from the point of view of intra-team interactions –

- "Hiding" much of the interface, even though not relevant to the team, can cause inefficiency and friction in the collaborative process. It may well be possible that revealing too little information as needed can cause lack of situational awareness while leaving it all out there is likely to cause cognitive overload. As such, there needs to be a delicate balance between how much of the shared GUI can be abstracted out into the mixed reality workspace.
- Allowing for a distributed workspace also requires processing of concurrent requests (for replanning, validation, etc.) which needs to be handled gracefully at the frontend – e.g. two commanders making concurrent edits on the public plan in their own mixed-reality spaces is undesirable and needs to be orchestrated effectively.

We hope to discuss some of these challenges, as well as report on our work in progress, at the workshop.

The system will be demonstrated live in the ICAPS-2018 System Demonstrations Track. An updated description of the system (presented at the ICAPS-2018 Workshop on User Interfaces and Scheduling and Planning) can be accessed online at http://rakaposhi.eas.asu.edu/ma_radar.pdf.

Acknowledgements. This research is supported in part by the AFOSR grant FA9550-18-1-0067, the ONR grants N00014-16-1-2892, N00014-13-1-0176, N00014-13-1-0519, N00014-15-1-2027, N00014-18-1-2442 and the NASA grant NNX17AD06G. Chakraborti is also supported by the IBM Ph.D. Fellowship for the years 2016-18.

References

Brafman, R. I., and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *ICAPS*, 28–35.

Brafman, R. I. 2015. A privacy preserving algorithm for multi-agent planning and search. In *IJCAI*, 1530–1536.

Chakraborti, T.; Sreedharan, S.; Kambhampati, S.; and Zhang, Y. 2017a. Explanation generation as model reconciliation in multi-model planning. Technical report.

Chakraborti, T.; Talamadupula, K.; Dholakia, M.; Srivastava, B.; Kephart, J. O.; and Bellamy, R. K. 2017b. Mr. Jones – Towards a Proactive Smart Room Orchestrator. *AAAI Fall Symposium on Human-Agent Groups*.

Karafantis, L. 2013. NASA's Control Centers: Design and History. Engineering and Technology.

Kim, J., and Shah, J. 2017. Towards intelligent decision support in human team planning. In *AAAI Fall Symposium on Human-Agent Groups*.

Klein, G. 2008. Naturalistic decision making. *The Journal of the Human Factors and Ergonomics Society*.

Murphy, M. 2015. Searching for Eureka: IBMs path back to greatness, and how it could change the world. Quartz.

Parasuraman, R.; Sheridan, T. B.; and Wickens, C. D. 2000. A model for types and levels of human interaction with automation. *Trans. Sys. Man Cyber. Part A*.

Sengupta, S.; Chakraborti, T.; Sreedharan; and Kambhampati, S. 2017. RADAR - A Proactive Decision Support System for Human-in-the-Loop Planning. In *ICAPS Workshop on User Interfaces for Scheduling and Planning (UISP)*.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2018. Handling Model Uncertainty and Multiplicity in Explanations as Model Reconciliation. In *ICAPS*.

Williams, T.; Szafir, D.; Chakraborti, T.; and Ben Amor, H. 2018. Virtual, augmented, and mixed reality for human-robot interaction. In *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 403–404. ACM.