# Human-Aware Goal-Oriented Autonomy through ROS-Integrated Timeline-based Planning and Execution

Alessandro Umbrico, Amedeo Cesta, and Andrea Orlandini

***Abstract*— Robots acting in real-world environments may interact with humans at different levels of abstraction (e.g., process, task, physical), entailing different control and coordination challenges. When acting in social situations, robots should be able to pursue (joint) goals by behaving according to the context as well as the skills/features of involved humans. Although reliable and effective, standard control techniques may limit the adaptability of robots. Novel control technologies based on Artificial Intelligence can endow robots with the cognitive capabilities needed to achieve a higher level of autonomy in terms of flexibility, reliability, and awareness. In this context, this paper introduces a goal-oriented acting framework based on timeline-based planning and execution. The framework is evaluated on a realistic Human-Robot Collaboration manufacturing scenario. Results show the capability of dealing with the uncontrollable dynamics of humans achieving effective and reliable collaborations.**

## I. INTRODUCTION

Robots interacting with humans in shared environments should implement behaviors that are efficient but also reliable, and acceptable [1]. It is crucial to consider a social and multi-agent perspective in order to fulfill the expectations of involved humans and possibly take advantage of their collaboration. Human-Robot Interaction (HRI) requires reasoning about the *non-functional qualities* of the implemented robot behaviors [2], [3] (e.g., acceptability, human awareness, safety). Nevertheless, humans constitute a source of *uncertainty* limiting a reliable "prediction" of their behaviors in terms of e.g., goals, beliefs, and intentions [4], [5]. This uncertainty raises control issues with respect to robot autonomy, affecting the way robots achieve goals [6].

Such issues are especially relevant in scenarios where robots and humans actively collaborate to achieve some common objective. Taking into account Human-Robot Collaboration (HRC), there is a wide literature [7], [8] investigating novel collaboration schema aiming at transforming a mechanical and "anonymous" cooperation into a real *symbiosis* between two "acting entities" that *benefit* from each other [9]. Advanced *cognitive capabilities* are necessary to enhance robot autonomy and guarantee adaptive and contextualized interactions with humans [10], [11]. Robots should integrate reasoning, representational, and interacting skills in order to effectively: (i) *communicate* with humans; (ii) *understand* production-related events and goals; (iii) *recognize* opportunities of actions and; (iv) (autonomously) *decide* which task to perform, how and when. AI technology, planning & execution in particular, can implement cognitive capabilities necessary to achieve the desired level of autonomy [12] e.g., *context awareness*, *decision making* and *problem solving*.

AI planning and execution features have been traditionally embedded in robotics as independent and stand-alone processes. A classical example is the three-layer architecture [13] where planning and execution are two loosely coupled processes, implemented with even radically different technologies. Planning and execution techniques that reduce the *representational gaps* between planning and robotics support the synthesis of plans that are effective and reliable in the real world [14]. A tighter integration of planning and execution is thus crucial to support *goal-oriented autonomy* [15].

This work addresses the autonomy problem by leveraging the timeline-based planning formalism [16] which supports reasoning on temporal dynamics and controllability issues [17]. Specifically, we introduce ROXANNE (ROs fleXible ActiNg coNtrollEr) as a novel planning and execution framework integrating timeline-based capabilities into the (standard) Robot Operating System (ROS). Other works in the literature propose planning-based controllers for ROS. ROSPlan [18] for example supports PDDL-based planners [19], [20] that do not consider *uncontrollable dynamics* of the environment. SkiROS [21] supports the flexible representation of robotic skills and their dynamic composition through task planning features. Still, the approach relies on PDDL-based (classical) planning without considering time and uncontrollable dynamics. The work FLEXHRC+ [22] also relies on PDDL but its control architecture follows a reactive perspective mainly. The framework makes choices on a short time horizon limiting the optimization capabilities necessary in practical scenarios (e.g., collaborative assembly).

Although interesting, these frameworks mainly rely on planning features that are not effective in dealing with the uncontrollable dynamics of human behaviors. ROXANNE overcomes these limitations by taking into account the execution perspective early at planning time. An explicit model of uncontrollable dynamics allows ROXANNE to synthesize plans that are robust when interacting with humans.

## II. ROXANNE

ROXANNE implements goal-oriented autonomy through the timeline-based approach as formalized in [16][1]. This formalism was introduced in the early 90s [23] and was successfully applied to real-world scenarios [24], [25], [26],

This work is partially supported by ROXANNE project (H2020, ROSIN – G.A. 732287).

CNR - Institute of Cognitive Sciences and Technologies, Rome, Italy {name.surname}@istc.cnr.it

[1]Due to space limitations, we invite readers to refer to [16] for an exhaustive description of the timeline-based formalism.

leading to the development of several frameworks e.g., EUROPA [27], T-REX [28], PLATINUm [29], oRatio [30].

A set of *state variables* characterizes temporal behaviors of domain features. A set of *synchronization rules* specifies global constraints coordinating the temporal evolution of state variables. While state variables specify local constraints describing the correct dynamics of each domain feature, synchronization rules coordinate state variables to achieve complex goals. *Timelines* are (planned) sequences of values instantiated over flexible temporal intervals called *tokens* and describe flexible behaviors of state variables. A key distinguishing feature is the modeling of the *uncontrollable dynamics* through *temporal uncertainty*. The formalism explicitly models *controllability properties* of state variable values as: (i) controllable; (ii) partially controllable; (iii) uncontrollable. Controllable values are actions whose execution is totally under the control of the agent. Partially controllable values are actions whose execution is affected by the environment (i.e., the agent decides when to start an action but can only observe its end). Uncontrollable values are actions whose execution can only be observed (i.e., the agent can only observe the start/end of the execution).

The acting process of ROXANNE is implemented through a dedicated ROS node. ROS topics allow the agent to send commands to the robotic platform, receive execution feedback, and monitor the state of the environment. Figure 1 shows the general structure of a ROXANNE acting node and its integration into ROS [2].

An acting node receives task requests on a dedicated input topic ("/roxanne/acting/task"). Requests are forwarded to the internal planning process which synthesizes a timeline-based plan. When a plan is ready then the executive starts its execution by interacting with the physical system through the *Platform Interface* (a *proxy* of the robot control layer). For each token to be executed, the executive dispatches a command through the output topic ("/roxanne/acting/dispatching") and receives the result from the feedback topic ("/roxanne/acting/feedback"). Note that ROXANNE can be configured with multiple pairs of dispatch-feedback topics, depending on the number of physical entities to be controlled/coordinated. Fully uncontrollable or "external" tokens are instead monitored through the observation topic ("/roxanne/acting/observation").

### A. Planning Under Temporal UIERIncertainty

The deliberative process of ROXANNE encapsulates the timeline-based planner PLATINUm [29] which synthesizes plans with minimal execution properties Taking into account temporal uncertainty, an interesting property is *dynamic controllability* which guarantees the successful execution in all possible evolution of uncontrollable features [17], [31]. ROXANNE focuses on *pseudo-controllability* which is a necessary condition for dynamic controllability. This property guarantees that the duration bounds of the uncontrollable and partially controllable tokens of a plan are *not altered* with respect to the domain specification of the associated values.

Algorithm 1 describes the pseudo-controllability aware planning procedure. It starts with a set of partially constrained timelines that are refined until all planning goals have been satisfied and a complete and valid set of timelines is found. Each iteration (rows 3-18) refines the current plan by detecting *flaws* (rows 4-12) and computing possible solutions (rows 15-18). Flaws represent conditions affecting the validity or completeness of the timelines. Each solution of a flaw determines an alternative refinement of a plan (*backtracking* point) and a new branch in the search space. The pseudo-controllability check (rows 4-5) verifies the constrained uncertainty of tokens with respect to the duration bounds of the domain specification. If the check fails, the procedure backtracks to an alternative plan since the current one would not be dynamically controllable (row 14),.

**Algorithm 1** Pseudo-controllability aware planning procedure

```
1:  function SOLVE_PC(P, S, H)
2:      π ← InitialPlan(P), F_pc ← ∅
3:      while ¬IsSolution(π) do
4:          U = {u_1, ..., u_n} ← GetUncertainty(π)
5:          if ¬Squeezed(U) then                ▷ Pseudo-controllability check
6:              Φ* = {φ*_1, ..., φ*_m} ← DetectFlaws(π, H)
7:              for φ*_i ∈ Φ* do                ▷ Compute flaw solutions
8:                  N_φ*i = {n_1, ..., n_t} ← HandleFlaw(φ*_i, π)
9:                  if N_φ*i = ∅ then
10:                     Backtrack(π, Dequeue(F_pc))
11:                 for n_j ∈ N_φ*i do          ▷ Branching for each solution
12:                     F_pc ← Enqueue(n_j, S)
13:         else
14:             Backtrack(π, Dequeue(F_pc))
15:         if ¬IsEmpty(F_pc) then              ▷ Iterative refinement
16:             π ← Refine(π, Dequeue(F_pc))
17:         else
18:             return Failure
19:     return π
```

### B. Controllability-Aware Execution

The execution process consists of a sequence of closed-loop control cycles discretizing the temporal axis into time units called *ticks* [32]. Each tick further consists of two phases. A *synchronization phase* verifies if the planned timelines comply with the observed state of the environment. A *dispatching phase* decides the tokens of timelines that can start their execution. Algorithm 2 describes the execution procedure which takes as input a plan $\Pi$ to be executed and a clock $\mathcal{C}$ generating ticks with a certain frequency.

**Algorithm 2** The executive procedure

```
1:  function EXECUTE(Π, C)
2:      π_exec ← Setup(Π)
3:      while ¬CanEndExecution(π_exec) do
4:          τ ← GetTick(C)                      ▷ Wait the next tick τ
5:          valid ← Synchronize(τ, π_exec)      ▷ Verify plan consistency
6:          if valid then                       ▷ Issue commands if π_exec is consistent
7:              Dispatch(τ, π_exec)
8:          else
9:              return Failure                  ▷ Trigger execution failure
```

[2] https://github.com/pstlab/roxanne_rosjava.git

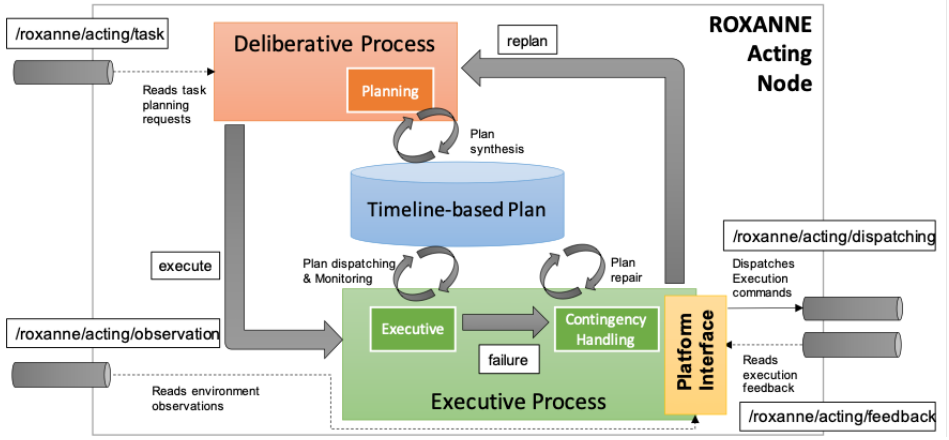



Fig. 1. Architecture of a ROXANNE planning and execution node.

Tokens follow different execution life cycles depending on their controllability properties. All tokens are set to the *waiting* state at the beginning of plan execution. Controllable tokens are set to the *in-execution* state as soon as all start conditions are met. They can then be set to the *executed* state as soon as all end conditions are satisfied. Similarly, partially-controllable tokens are set to the *in-execution* state as soon as all start conditions are met. However, the executive waits for (positive) feedback from the environment before setting these tokens to the *executed* state. In the case of uncontrollable tokens, feedback from the environment is necessary even to start their execution. The executive would thus set such tokens to the *in-execution* state and *executed* state only after receiving proper signals/feedback from the environment.

### C. Execution Failures

Algorithm 3 describes with more detail the synchronization phase. The procedure collects feedback and observations received until the current tick $\tau$ in order to verify the consistency of the plan with respect to the actual state of the environment (rows 2-12). Observed temporal information about tokens are propagated into the plan and checked for consistency (rows 8, 10, and 11). If propagation fails, the plan does not comply with the environment and the procedure triggers a *failure* (rows 11-12).

**Algorithm 3** The synchronization procedure

```
1:  function SYNCHRONIZE(τ, π_exec)
2:      O = {o_1, ..., o_n} ← GetObservations(τ, π_exec)
3:      for o_i ∈ O do                          ▷ Check observations received at tick τ
4:          if isFeedback(o_i) then             ▷ Verify execution feedback
5:              if isError(o_i) then            ▷ Execution error
6:                  return Failure
7:              else                            ▷ Successful execution
8:                  π_exec ← PropagateEndTime(τ, o_i)
9:          else                                ▷ Verify observation
10:             π_exec ← PropagateObservation(τ, o_i)
11:     if ¬ IsConsistent(π_exec) then
12:         return Failure
13:     A = {a_i, ..., a_m} ← GetControllableActions(π_exec)
14:     for a_i ∈ A do                          ▷ Check controllable actions
15:         if CanEndExecution(τ, a_i, π_exec) then
16:             π_exec ← PropagateEndTime(τ, a_i)
```

Failures correlated with execution feedback concerns *errors* (e.g., a dispatched command was not executed due to some engine failure) or *asynchronism* (e.g., a dispatched command did not end within the expected time). When a failure is detected, *plan adaptation* is necessary to achieve the agent goal. The *Contingency Handler* in this case is in charge of restoring a stable state of the plan before re-planning. Specifically, it waits for feedback about tokens in execution and stops controllable tokens as soon as possible. When no token is being executed then a new plan can be synthesized by adapting the partially executed timelines.

## III. ROBOT ACTING WITH HUMANS

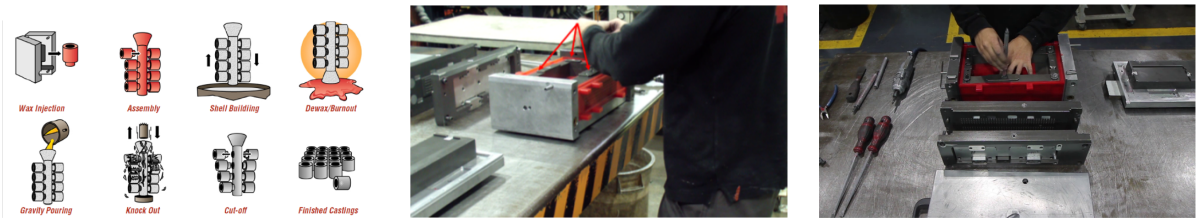
Human-Robot Interaction is well suited to assess the capabilities of ROXANNE. Scenarios entailing the tight interaction between a controllable autonomous agent (i.e., the robot) with an uncontrollable autonomous agent (i.e., the human) are challenging. We in particular focus on Human-Robot Collaboration (HRC) where a robot and a human jointly carry out production processes. A key challenge of such scenarios is the efficient coordination of the human and the robot (i.e., maximizing production throughput) while supporting reliability and safety.

### A. Collaborative Assembly

We consider a realistic industrial scenario inspired by the H2020 research project FourByThree[3] [33]). The process requires a human and a robot to jointly assemble/disassemble a metal die used to produce a wax part through an injection machine. Figure 2(a) shows the phases of the whole process. The collaboration focuses on the preparation of the die for the removal of the wax part, see Figure 2(b) and Figure 2(c).

The collaborative process consists of four macro steps executed in strict order. The first macro step loads the metal die on the shared workspace (*set workpiece*) and can be performed by the human only. The second macro step is *remove top cover* and consists of a variable number of *unscrewing tasks* performed by the human and/or the

[3] https://fourbythree.eu




(a) (b) (c)

Fig. 2. Overview of the scenario: (a) phases of the whole process; (b) and (c) assembly steps considered for the introduction of a collaborative robot.

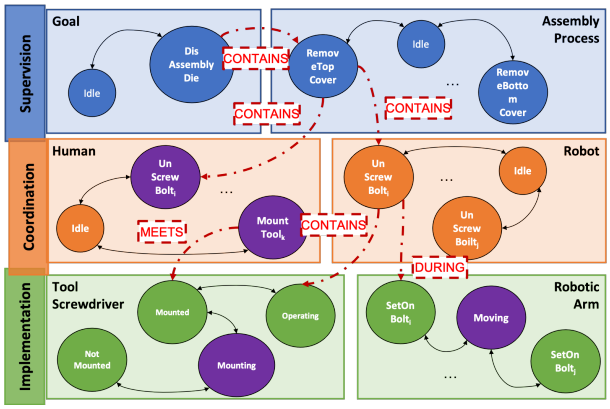



Fig. 3. Hierarchical domain specification.

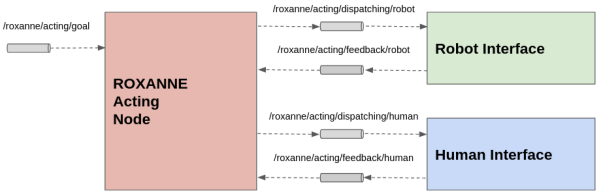



Fig. 4. Deployment of the ROXANNE acting node

robot. The third macro step is *remove wax part* and can be performed by the human only. The last macro step then is the *remove bottom cover* and consists of several *unscrewing tasks* performed by both agents.

Figure 3 shows the structure of the timeline-based model organized into three abstraction levels. The *supervision level* concerns state variables that characterize high-level goals and related production tasks. The *coordination level* concerns state variables that characterize the low-level tasks (e.g., unscrewing tasks) the human and the robot can perform. The *implementation level* concerns state variables that characterize primitive actions the robot would execute to implement assigned tasks. Dotted (red) arrows describe temporal relationships constraining the procedural implementation of modeled tasks (i.e., correlate values of state variables at different levels of abstraction).

The human is modeled as an *uncontrollable* entity of the environment while the robot is *partially controllable*. The values representing the low-level tasks the human can perform are thus modeled as *uncontrollable*. The values representing the low-level tasks the robot can perform are modeled as *partially-controllable*. The behavior of the human can affect the execution of robot tasks (e.g., slow-down robot motions for safety reasons) thus the robot can decide when to start a task but not its end.

### B. Deploying ROXANNE

ROXANNE is open-source and is developed as a ROS standard package. A *ROXANNE configuration service* ("/roxanne/acting/configuration") initializes and configures an acting node on a given model. A *configuration file* declares the set of topics used by the acting agent to exchange information with other ROS nodes. Figure 4 schematically describes the deployment of ROXANNE in the considered HRC scenario. Two distinct pairs of dispatching-feedback topics are defined in order to support the simultaneous execution of human and robot tasks. The human interface component is usually a Human-Machine-Interface (HMI) supporting communication to and from the worker. The components *RobotInterface* and *HumanInterface* are two ROS nodes simulating partially controllable and uncontrollable execution of tokens dispatched respectively to the robot and the human.

## IV. EXPERIMENTAL RESULTS

The experiments evaluate the *human awareness* of ROXANNE at both planning and execution levels. From the planning point of view, experiments show the capabilities of planning effective collaborations, taking into account temporal uncertainty. From the execution point of view, experiments show the capability of reliably carrying out robot tasks dealing with the uncontrollable behavior of humans. In particular, the experiments clearly show that a correct estimation of the temporal uncertainty allows ROXANNE to reliably deal with execution variability (i.e., limit replanning) while guaranteeing a good level of optimization.

### A. Assess Human-Awareness at Planning Level

Several domain specifications and problem instances have been defined by varying the following parameters: (i) the number of tasks that must be carried out to complete the assembly (10, 15, 20, 25, and 30 tasks); (ii) the number of *shared tasks* that can be assigned to the robot or to the human (20%, 40%, 60%, 80%, and 100% of shared tasks); (iii) the amount of temporal uncertainty concerning the duration of human tasks ($\pm$ 10, $\pm$ 20, and $\pm$ 30 uncertainty on average task duration). The steps where human-robot collaboration can actually occur are the *remove top cover* and the *remove*

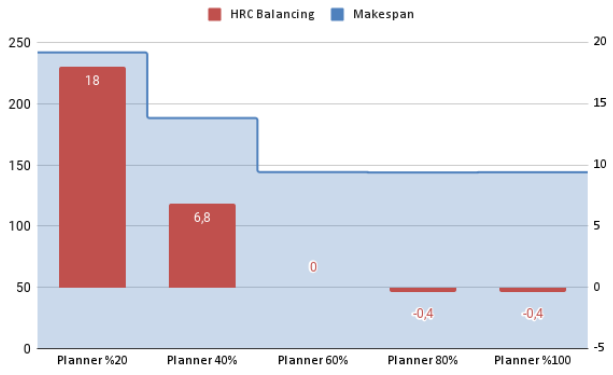



Fig. 5. Comparison of plan makespan and task allocation distribution.

*bottom cover*. The number of the needed unscrewing tasks varies from a minimum of 10 tasks to a maximum of 30 tasks. The number of underlying shared (unscrewing) tasks then varies from a minimum of 20% to a maximum of 100%.

ROXANNE has been run on 75 versions of the planning domain with a solving timeout set to 3 minutes [4]. The planner has been configured with a hierarchical flaw selection heuristics introduced in [34] and a dedicated search strategy minimizing the makespan of the collaborative process and the workload distribution.

Results show that the planner scales well with the increasing size of the problem ranging from a minimum average time of $5,96$ seconds ($\delta = 6,52$) for 20% of shared tasks, to a maximum average time of $39,66$ seconds ($\delta = 46,56$) for 100% of shared tasks. Intuitively, the higher the number of tasks the higher the decisions made during solving. The higher the number of shared tasks the larger the set of alternative plans evaluated during the search. The amount of uncertainty instead does not affect planning time but is important with respect to the robustness of generated plans (more details in the evaluation of the execution).

More interestingly, Figure 5 shows the qualities of the plans. The bar chart shows the average distribution of the workload (HRC Balancing) pointing out the average number of tasks assigned to the human (positive numbers) and to the robot (negative numbers). The line chart instead shows the average makespan of the resulting collaborative processes. A higher number of shared tasks (i.e., planning decisions) allows ROXANNE to better distribute tasks and increase productivity through a lower makespan.

### B. Assess Human-Awareness at Execution Level

To assess the execution capabilities of ROXANNE under different temporal dynamics, we have simulated several collaborative scenarios taking into account a fixed number of planned tasks (10), a fixed number of shared tasks (60%), and a variable amount of temporal uncertainty. Uncontrollable dynamics of human and robot tasks are simulated through a developed *platform simulator* which randomly generates execution feedback with increasing time variability. The experiments vary *model uncertainty* and *platform uncertainty*. The former characterizes the temporal uncertainty modeled into the timeline-based specification (values: 10, 20, and 30). The latter characterizes the actual variability of the platform simulator when triggering execution feedback (values: 5, 10, 15, 20, 25, and 30).

Figure 6 compares the observed acting behavior of ROXANNE under different combinations of model uncertainty and platform uncertainty. The bar charts show the number of failures detected during execution. The line charts show the time taken to complete the execution of the plans. Failures concern the uncontrollable duration of tasks. Namely, failures were detected every time the observed duration of dispatched tasks did not comply with the modeled uncertainty. Each failure triggers some re-planning in order to reconcile the plan with the actual state of the environment and complete execution. Clearly, the higher the number of failures detected, the higher the number of re-planning sessions leading to lesser efficient collaborations. This is especially evident in Figure 6(a) where failures led to the longest executions.

Failures are generally detected when the temporal uncertainty of the planning model is too optimistic with respect to the actual uncertainty of the environment. For example, according to Figure 6(a) ROXANNE starts detecting failures with platform uncertainty 15 when the temporal uncertainty of the model is 10. Similarly, according to Figure 6(b), ROXANNE starts detecting failures with platform uncertainty 25 when the temporal uncertainty of the model is 20. Figure 6(c) instead shows that no failure is detected when the temporal uncertainty of the model (30) is higher than or equal to the actual uncertainty of the environment. Pessimistic models of uncertainty lead to reliable acting behaviors.

## V. FINAL REMARKS AND FUTURE WORKS

This paper presents a ROS-integrated timeline-based framework supporting human-aware goal-oriented autonomy. The underlying timeline-based formalism [16] and the embedded planning engine PLATINUm [29] support representation and reasoning on *temporal uncertainty*. Temporal uncertainty is paramount to take into account the execution perspective early at planning time and realize reliable acting behaviors. Experimental results on a simulated but realistic collaborative scenario show the efficacy of ROXANNE in synthesizing efficient and robust acting behaviors in highly variable situations. Future works will enhance the execution capabilities of ROXANNE by reasoning on plan knowledge to make better-dispatching decisions and support early detection of failures (e.g., anticipate failures before receiving feedback when known conditions are observed).

[4]The experiments have been run on a MacBook Pro with processor Intel Core i5 2.4 GHz and 8 GB of memory.

## REFERENCES

[1] S. Rossi, F. Ferland, and A. Tapus, "User profiling and behavioral adaptation for HRI: A survey," *Pattern Recognition Letters*, vol. 99, pp. 3 – 12, 2017.

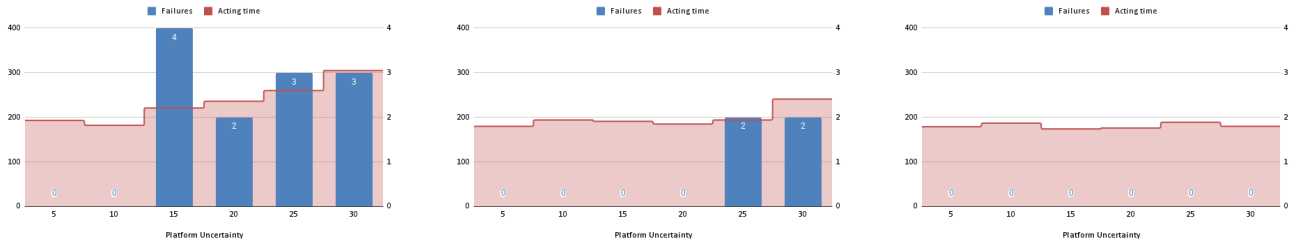



Fig. 6. Comparison of acting behaviors under different models of temporal uncertainty: (a) 10; (b) 20; (c) 30.


[2] B. Bruno, C. T. Recchiuto, I. Papadopoulos, A. Saffiotti, C. Koulouglioti, R. Menicatti, F. Mastrogiovanni, R. Zaccaria, and A. Sgorbissa, "Knowledge representation for culturally competent personal robots: Requirements, design principles, implementation, and assessment," *International Journal of Social Robotics*, vol. 11, no. 3, pp. 515–538, 2019.

[3] I. Awaad, G. K. Kraetzschmar, and J. Hertzberg, "The Role of Functional Affordances in Socializing Robots," *International Journal of Social Robotics*, vol. 7, no. 4, pp. 421–438, 2015.

[4] A. Clodic and R. Alami, *What Is It to Implement a Human-Robot Joint Action?* Cham: Springer International Publishing, 2021, pp. 229–238.

[5] A. Clodic, E. Pacherie, R. Alami, and R. Chatila, *Key Elements for Human-Robot Joint Action*. Cham: Springer International Publishing, 2017, pp. 159–177.

[6] M. Selvaggio, M. Cognetti, S. Nikolaidis, S. Ivaldi, and B. Siciliano, "Autonomy in physical human-robot interaction: A brief survey," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7989–7996, 2021.

[7] G. Michalos, P. Karagiannis, N. Dimitropoulos, D. Andronas, and S. Makris, *Human Robot Collaboration in Industrial Environments*. Cham: Springer International Publishing, 2022, pp. 17–39.

[8] S. Makris, *Cooperating Robots for Flexible Manufacturing*, Springer, Ed. Springer International Publishing, 2021, no. 1.

[9] D. Romero, P. Bernus, O. Noran, J. Stahre, and Å. Fast-Berglund, "The operator 4.0: Human cyber-physical systems & adaptive automation towards human-automation symbiosis work systems," in *Advances in Production Management Systems. Initiatives for a Sustainable World*, I. Nääs, O. Vendrametto, J. Mendes Reis, R. F. Gonçalves, M. T. Silva, G. von Cieminski, and D. Kiritsis, Eds. Cham: Springer International Publishing, 2016, pp. 677–686.

[10] A. Andriella, C. Torras, and G. Alenyà, "Short-term human–robot interaction adaptability in real-world environments," *International Journal of Social Robotics*, vol. 12, no. 3, pp. 639–657, Jul 2020.

[11] A. Umbrico, A. Cesta, G. Cortellessa, and A. Orlandini, "A holistic approach to behavior adaptation for socially assistive robots," *International Journal of Social Robotics*, vol. 12, no. 3, pp. 617–637, Jul 2020.

[12] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol. 247, pp. 10 – 44, 2017, special Issue on AI and Robotics.

[13] E. Gat, R. P. Bonnasso, R. Murphy, *et al.*, "On three-layer architectures," *Artificial intelligence and mobile robots*, vol. 195, p. 210, 1998.

[14] K. Rajan and A. Saffiotti, "Towards a science of integrated ai and robotics," *Artificial Intelligence*, vol. 247, pp. 1 – 9, 2017, special Issue on AI and Robotics.

[15] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy," *The International Journal of Robotics Research*, vol. 17, no. 4, pp. 315–337, 1998.

[16] M. Cialdea Mayer, A. Orlandini, and A. Umbrico, "Planning and execution with flexible timelines: a formal account," *Acta Informatica*, vol. 53, no. 6, pp. 649–680, 2016.

[17] T. Vidal and H. Fargier, "Handling Contingency in Temporal Constraint Networks: From Consistency To Controllabilities," *JETAI*, vol. 11, no. 1, pp. 23–45, 1999.

[18] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, and M. Carreras, "Rosplan: Planning in the robot operating system," in *Twenty-Fifth International Conference on Automated Planning and Scheduling*, 2015.

[19] D. Mcdermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL - The Planning Domain Definition Language," Tech. Rep., 1998.

[20] M. Fox and D. Long, "PDDL2.1: an extension to PDDL for expressing temporal planning domains," *J. Artif. Intell. Res. (JAIR)*, vol. 20, pp. 61–124, 2003.

[21] F. Rovida, M. Crosby, D. Holz, A. S. Polydoros, B. Großmann, R. P. A. Petrick, and V. Krüger, *SkiROS—A Skill-Based Robot Control Platform on Top of ROS*. Cham: Springer International Publishing, 2017, pp. 121–160.

[22] K. Darvish, E. Simetti, F. Mastrogiovanni, and G. Casalino, "A hierarchical architecture for human–robot cooperation processes," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 567–586, 2021.

[23] N. Muscettola, "HSTS: Integrating Planning and Scheduling," in *Intelligent Scheduling*, Zweben, M. and Fox, M.S., Ed. Morgan Kauffmann, 1994.

[24] A. Cesta, G. Cortellessa, S. Fratini, and A. Oddi, "Developing an End-to-End Planning Application from a Timeline Representation Framework," in *IAAI-09. Proc. of the 21$^{st}$ Innovative Application of Artificial Intelligence Conference, Pasadena, CA, USA*, 2009.

[25] A. Jonsson, P. Morris, N. Muscettola, K. Rajan, and B. Smith, "Planning in Interplanetary Space: Theory and Practice," in *AIPS-00. Proceedings of the Fifth Int. Conf. on AI Planning and Scheduling*, 2000.

[26] S. Pellegrinelli, A. Orlandini, N. Pedrocchi, A. Umbrico, and T. Tolio, "Motion planning and scheduling for human and industrial-robot collaboration," *CIRP Annals*, vol. 66, no. 1, pp. 1–4, 2017.

[27] J. Barreiro, M. Boyce, M. Do, J. Frank, M. Iatauro, T. Kichkaylo, P. Morris, J. Ong, E. Remolina, T. Smith, and D. Smith, "EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization," in *ICKEPS 2012: the 4th Int. Competition on Knowledge Engineering for Planning and Scheduling*, 2012.

[28] F. Py, K. Rajan, and C. McGann, "A Systematic Agent Framework for Situated Autonomous Systems," in *AAMAS-10. Proc. of the 9$^{th}$ Int. Conf. on Autonomous Agents and Multiagent Systems*, 2010.

[29] A. Umbrico, A. Cesta, M. Cialdea Mayer, and A. Orlandini, "Platinum: A new framework for planning and acting," in *AI*IA 2017 Advances in Artificial Intelligence*, F. Esposito, R. Basili, S. Ferilli, and F. A. Lisi, Eds. Cham: Springer International Publishing, 2017, pp. 498–512.

[30] R. De Benedictis and A. Cesta, "Lifted heuristics for timeline-based planning," in *ECAI 2020 - 24th European Conference on Artificial Intelligence*, ser. Frontiers in Artificial Intelligence and Applications, G. D. Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín, and J. Lang, Eds., vol. 325. IOS Press, 2020, pp. 2330–2337.

[31] S. Bensalem, K. Havelund, and A. Orlandini, "Verification and validation meet planning and scheduling," *International Journal on Software Tools for Technology Transfer*, vol. 16, no. 1, pp. 1–12, Feb 2014.

[32] F. Py, K. Rajan, and C. McGann, "A systematic agent framework for situated autonomous systems," in *AAMAS*, 2010, pp. 583–590.

[33] I. Maurtua, N. Pedrocchi, A. Orlandini, J. d. G. Fernández, C. Vogel, A. Geenen, K. G. LarsenAlthoefer, and A. Shafti, "Fourbythree: Imagine humans and robots working hand in hand," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016, pp. 1–8.

[34] A. Umbrico, A. Orlandini, and M. Cialdea Mayer, "Enriching a temporal planner with resources and a hierarchy-based heuristic," in *AI*IA 2015, Advances in Artificial Intelligence*. Springer International Publishing, 2015, pp. 410–423.