# PLATINUM
# A new Framework for Planning and Acting

Alessandro Umbrico[1], Amedeo Cesta[1], Marta Cialdea Mayer[2], Andrea Orlandini[1]

[1] Istituto di Scienze e Tecnologie della Cognizione
Consiglio Nazionale delle Ricerche, Roma
[2] Dipartimento di Ingegneria
Università degli Studi Roma Tre

**Abstract.** This paper presents a novel planning framework, called PLATINUM that advances the state of the art with the ability of dealing with temporal uncertainty both at planning and plan execution level. PLATINUM is a comprehensive planning system endowed with (i) a new algorithm for temporal planning with uncertainty, (ii) heuristic search capabilities grounded on hierarchical modelling and (iii) a robust plan execution module to address temporal uncertainty while executing plans. The paper surveys the capabilities of this new planning system that has been recently deployed in a manufacturing scenario to support Human-Robot Collaboration.

## 1 Introduction

The continuous improvements in robotics in terms of efficacy, reliability and costs are fostering a fast diffusion in a large variety of scenarios where *robots* are required to demonstrate more flexible and interactive features like, e.g., those for supporting and interacting with humans. For instance, during the last decade *lightweights robots* are being increasingly used in manufacturing cells to support human workers in repetitive and physical demanding operations. The co-presence of a robot and a human in a shared environment while operating actively together entails many issues that must be properly addressed requiring the deployment of *flexible controllers* capable of preserving *effectiveness* while enforcing *human safety*. In manufacturing, Human-Robot Collaboration (HRC) challenges concern both *physical interactions*, guaranteeing the *safety* of the human, and *coordination* of activities, improving the productivity of cells [7].

In such scenarios, the presence of a human, which plays the role of an *uncontrollable* "agent" in the environment, entails the deployment of control systems capable of evaluating *online* the robot execution time and continuously adapt its behaviors. Namely, deliberative control systems are required to leverage temporal flexible models (such as in [6]) as a key enabling feature both at planning and execution time. In this sense, standard methods are not fully effective as current approaches do not foresee/estimate the actual time needed by robots to perform collaborative tasks (i.e., tasks that directly or indirectly involve humans). Indeed, robot trajectories are usually computed *online* by taking into account the current position of the human and, therefore, it is not possible to know in advance the time the robot will need to complete a task. Thus, it is

not possible to plan robot and human tasks within a long production process and take into account performance issues at the same time.

Some plan-based controllers rely on temporal planning mechanisms capable of dealing with coordinated task actions and temporal flexibility e.g., T-REX [17] or IXTET-EXEC [10] that rely respectively on EUROPA [1] and IXTET [8] temporal planners. It is worth noting how both these systems do not have an explicit representation of *uncontrollability* features in the planning domain. As a consequence, the resulting controllers are not endowed with the *robustness* needed to cope with *uncontrollable dynamics* of domains such as, for instance, the one needed in HRC scenarios.

This paper presents a new Planning framework, called PLATINUM, which integrates temporal planning and execution capabilities that both explicitly deal with temporal uncertainty, thus resulting as well tailored for flexible human-robot collaborative scenarios. The system has been developed and deployed within the FOURBYTHREE research project[3] [12]. The PLATINUM planning and acting capabilities have been integrated in a software environment that facilitates the adaptation of a new robotic arm in different HRC manufacturing scenarios. The proposed planning system has been completely deployed in a realistic case study [16] demonstrating its ability to support a productive and safe collaboration between human and robot. In particular, PLATINUM has been able to find well suited task distribution between human and robot increasing the productivity of the working cell, without affecting the safety of the operator.

## 2 Human-Robot Collaboration: needs from a case study

In manufacturing, HRC scenarios consist of a human operator and a robot that interact and cooperate to perform some common tasks. Namely, the human and the robot represent two *autonomous agents* capable of performing tasks, affecting each other behaviors and sharing the same working environment.

The motivations of this work rely on a research initiative related to the FOURBYTHREE project funded by the European Commission. FOURBYTHREE is a research project [12] whose main aim is to realize new robotic solutions that allow human operators to safely and efficiently collaborate with robots in manufacturing contexts. Specifically, the project outcomes will be a new generation of collaborative robotic solutions based on innovative hardware and software. The envisaged solutions present four main characteristics (*modularity*, *safety*, *usability* and *efficiency*) and take into account the co-presence of three different actors (the *human*, the *robot* and the *environment*). In this context, the solution proposed by FOURBYTHREE is a combination of several hardware and software components for implementing safe and effective HRC applications. On the one hand, a brand new collaborative robotic arm has been designed and is under validation. On the other hand, a set of software modules spanning from very high level features, such as, e.g., voice and gesture commands detection, to low-level robot control have been developed. The resulting complex integrated robotic solution implements two possible robot-human relationships in a given workplace without physical fences:

[3] CNR authors are partially supported by EU project FOURBYTHREE (GA No.637095 – http://www.fourbythree.eu).

(i) *coexistence* (the human and the robot conduct independent activities); (ii) *collaboration* (the human and the robot work collaboratively to achieve a shared productive goal). Validation tests are ongoing in four pilot plants, covering different types of production process, i.e., assembly/disassembly, welding operations, large parts management and machine tending.

In this paper, one of the pilots in FOURBYTHREE is considered as a relevant HRC scenario for manufacturing. In such a scenario a robot and a human must cooperate in the assembly/disassembly of metal dies for the production of wax patterns. Some tasks of the process can be performed by both the robot and the human while other tasks that require a special dexterity can be performed only by the human. The robot is endowed with a screwdriver and therefore it can supports the human in all the screwing/unscrewing operations of the process. Such HRC scenario can be addressed deploying Planning and Scheduling (P&S) technology [4], i.e., modeling the control problem as a time-flexible planning problem and, then, solving it by means of a hierarchical timeline-based application [20]. The hierarchical approach provides a description of the problem at different levels of abstraction ranging from the process definition level to the robot task implementation level. Then, timelines coordinate the behaviors of the human and the robot over time in order to achieve the desired production goals.

### 2.1 Planning with Timelines under Temporal Uncertainty

The formal characterization of the timeline-based approach defined in [6] is well-suited to model HRC scenarios thanks to its capability of representing *temporal uncertainty*. Indeed, temporal uncertainty plays a relevant role in HRC where the human represents an *uncontrollable* element of the environment with respect to the robot. Thus, it is crucial to properly represent and handle such uncertainty in order to produce *robust* plans and dynamically adapt the behavior of the robot to the *observed* behavior of the human. According to [6], a domain specification is composed by a set of *multi-valued state variables*. Each state variable models the allowed temporal behaviors of a particular feature of the domain that must be controlled over time. A state variable is formally defined by the tuple $(V, D, T, \gamma)$ where: (i) $V$ is a set of values the feature can assume over time; (ii) $D : V \rightarrow \mathbb{R}_{>=0} \times \mathbb{R} \cup \infty$ is a duration function specifying for each value the allowed non negative minimum and maximum duration; (iii) $T : V \rightarrow 2^V$ is a transition function specifying the allowed sequences of values over the timeline; (iv) $\gamma : V \rightarrow \{c, u\}$ is a controllability tagging function specifying for each value whether it is controllable or not. State variables model a single feature of a domain by describing the local constraints that must be satisfied in order to build valid temporal behaviors (i.e., valid timelines). *Synchronization rules* specify additional (temporal) constraints that coordinate the behavior of different state variables in order to realize complex tasks or achieve goals.

For instance, Figure 1 partially shows the domain specification for the collaborative assembly/disassembly process in FOURBYTHREE. An *Assembly Process* state variable models the high-level tasks that must be performed in order to carry out the desired collaborative process. A *Robot* and a *Human* state variables model the possible behaviors of the robot and the human in terms of the low-level tasks they can perform over time. Finally, *Arm* and *Screwdriver* state variables model respectively robot motion tasks and

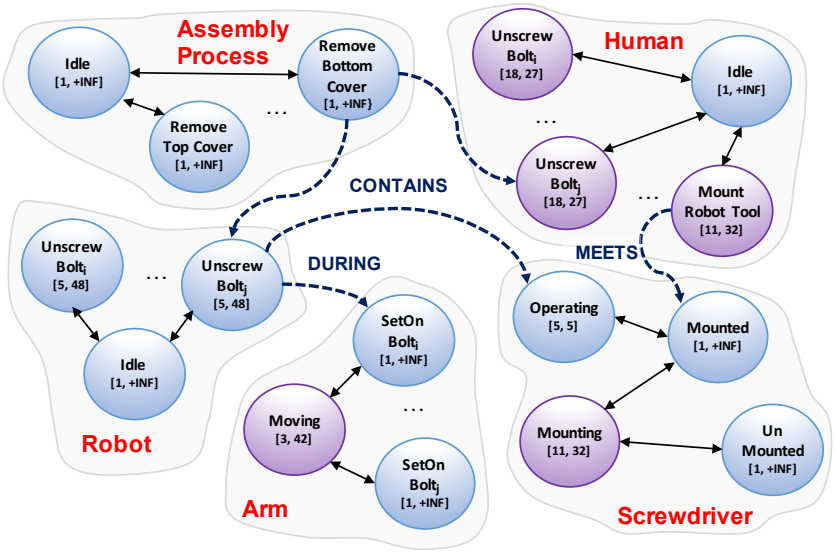


Fig. 1: A partial timeline-based domain specification for a HRC scenario

tool management. It is worth to underscore the capability of the formal framework to allow the modeling of uncontrollable dynamics in the considered scenario. The human agent is modeled as an element of the environment and therefore all human tasks are tagged as uncontrollable. In addition, a human may indirectly affect the behavior of the robot in such a working scenario because the robot can slow-down or even interrupt motion tasks in order to guarantee free-collision trajectories.Thus, the actual duration of motion tasks is not under the control of the control system and, therefore, such tasks are also tagged as uncontrollable. The dotted arrows in Figure 1 represent temporal constraints entailed by a synchronization rule defined for the value *Remove Bottom Cover* of *Assembly Process* state variable. Such constraints specify operational requirements needed to carry out a high-level task (*Remove Bottom Cover*) of the assembly/disassembly process. A set of *contains* temporal constraints specify the low-level tasks the human and the robot must perform. Specifically, they specify the bolts the human and the robot must unscrew to remove the cover. Then, additional temporal constraints (*during* and *contains* temporal constraints) specify how the robot must implement related low-level tasks. To successfully unscrew a bolt the robot arm must be set on the related location and the screwdriver must be activated.

In such a context, a *plan* is composed by a set of *timelines* and a set of *temporal relations* that satisfy the domain specification and achieve the desired goals. Each timeline is composed by a set of flexible temporal intervals, called *tokens* describing the possible temporal behavior of the related feature of the domain. *Temporal flexibility* allows timelines to encapsulate an envelope of possible temporal behaviors. Such a rich temporal representation together with controllability information can be exploited to generate plans that can be dynamically adapted to the observed dynamics of the environment at execution time.

### 2.2 A Hierarchical Modeling Approach for HRC scenarios

In general, the design of effective models is a crucial issue in the development of plan-based controllers. Indeed, a planning model must capture the information about the system to be controlled and the environment in which it works. A planning model is to capture such complexity and to allow a Planner to make decisions at different levels of abstraction. To this aim, hierarchical modeling approaches have been successfully applied in real world scenarios. They support the planning process by *encoding* knowledge about a particular problem to address. HRC scenarios are complex problems that require to take into account several aspects from different perspectives. Thus, hierarchies are well-suited in such a context as they allow to model a complex problem from different levels of abstraction and decompose the related complexity in sub-problems.

Pursuing the hierarchical modeling approach described in [20, 4], it is possible to model a HRC scenario by identifying three hierarchical levels. A *supervision level* models the process and the high-level tasks that must be performed. At this level of abstraction the model specifies the operational constraints that must be satisfied to carry out the process regardless of the *agent* that will perform the tasks. The state variable *Assembly Process* in Figure 1 is the result of such a level. A *coordination level* models the decomposition of high-level tasks of the process into low-level tasks that the human or the robot can directly handle. At this level of abstraction the model specifies the possible assignments of tasks to the robot and therefore the possible interactions between a human and a robot. The state variables *Human* and *Robot* in Figure 1 model the low-level tasks that the robot and the human may perform and the synchronization rules connecting these values with state variable *Assembly Process* model possible assignments. An *implementation level* models the operations and the related requirements that allow a robot to perform assigned tasks. The state variables *Arm* and *Screwdriver* in Figure 1 and the synchronization rules connecting their values with the state variable *Robot* model the motion tasks and the tool activations needed to carry out robot tasks.

## 3 A Framework for Planning & Execution under Uncertainty

The modeling and solving approaches described above have been implemented in a general framework called EPSL (*Extensible Planning and Scheduling Library*) [20]. EPSL complies with the formal characterization given in [6] and initial steps have been done in order to compare it with other state-of-the-art frameworks [19]. Nevertheless, EPSL was not fully suited to address the needs related to task planning problems in FOURBYTHREE. Most importantly, handling temporal uncertainty both at planning and execution time results as a key capability for effectively and safely deploy P&S robot control solutions. Thus, a new system, called PLanning and Acting with TImeliNes under Uncertainty (PLATINUM), is presented here constituting a uniform framework for planning and execution with timelines with (temporal) uncertainty[4].

The capability of handling temporal uncertainty both at planning and execution time allows the framework to address problems where not all the features of a domain are under the control of the system. Moreover, the "combination" of temporal flexibility

[4] https://github.com/pstlab/PLATINUm.git

and temporal uncertainty allows P&S controllers to generate *flexible* and *temporally robust* plans that can be dynamically adapted at execution time without generating new plans from scratch. Robust plan execution is particularly relevant in HRC scenarios in order to avoid to continuously generate a new plan every time an unexpected behavior of the human is detected (e.g., human execution delays).

Recent results [16] have shown the capability of a PLATINUM instance to realize flexible collaborations by dynamically adapting the behavior of a robot to the observed/detected behavior of a human. Before describing the deployment of PLATINUM in a realistic collaborative assembly scenario, next sections provide a description of how the P&S framework has been extended in order to deal with temporal uncertainty at both planning and execution time.

### 3.1 Solving Timeline-based Problems with Uncertainty

Given a domain specification and a particular problem to solve, the role of a timeline-based planner is to synthesize a set of flexible timelines that satisfy domain constraints and achieve some goals. A plan includes a set of timelines each of which describes the allowed temporal behaviors of a particular domain feature (i.e., state variables). In such a context, a plan represents an envelope of possible solutions. Indeed, *temporal flexibility* allows timelines to encapsulate an envelope of possible temporal behaviors. Given the considered HRC scenario, a plan consists of a set of coordinated human and robot beahviors that carry out a particular production task. The solving process of a P&S application can be generalized as a plan refinement search. Basically, a solver iteratively refines an initial partial plan until a valid and complete plan is found. The refinement of a plan consists in detecting and solving a set of *flaws* that affect either the validity or the completeness of the plan. However, the *validity* of a plan with respect to the domain specification does not represent a sufficient condition to guarantee its *executability* in the real world. The *uncontrollable dynamics* of the environment may prevent the complete and correct execution of plans. Thus, from the planning perspective, it is important to generate plans with some properties with respect to the *controllability problem* [13, 21]. *Dynamic controllability* is the most relevant property with respect to the execution of a plan in the real world. Unfortunately, it is not easy to deal with such a property at planning time when the temporal behaviors of domain features are not complete. Typically, such a property is taken into account with post-processing mechanisms after plan generation [5, 3, 13, 21]. Another property worth to be considered at planning time, is the *pseudo-controllability* property which represents a necessary (but not sufficient) condition for dynamic controllability [13].

The pseudo-controllability property of a plan aims at verifying that the planning process does not make hypotheses on the actual duration of the uncontrollable activities of a plan. Specifically, pseudo-controllability verifies that the planning process does not *reduce* the duration of uncontrollable values of the domain. Consequently, a timeline-based plan is pseudo-controllable if and only if all the flexible durations of uncontrollable tokens composing the timelines have not been changed with respect to the domain specification. Although pseudo-controllability does not convey enough information to assert the dynamic controllability of a plan, it represents a useful property that can be

exploited for *validating* the planning domain with respect to temporal uncertainty. Indeed, if the planner cannot generate pseudo-controllable plans, then it cannot generate dynamically controllable plans either. Thus, the general solving procedure of EPSL has been now extended in PLATINUM aiming at dealing with temporal uncertainty at planning time. Algorithm 1 shows the new PLATINUM planning procedure.

**Algorithm 1** A general pseudo-controllability aware planning procedure

```
1: function SOLVE(P, S, H)
2:     F_pc, F_¬pc ← ∅
3:     π ← InitialPlan(P)
4:     // check if the current plan is complete and flaw-free
5:     while ¬IsSolution(π) do
6:         // get uncontrollable values of the plan
7:         U = {u_1, ..., u_n} ← GetUncertainty(π)
8:         // check durations of uncontrollable values
9:         if ¬Squeezed(U) then
10:            // detect the flaws of the current plan
11:            Φ^0 = {φ_1, ..., φ_k} ← DetectFlaws(π)
12:            // apply the heuristic to filter detected flaws
13:            Φ* = {φ*_1, ..., φ*_m} ← SelectFlaws(Φ^0, H)
14:            // compute possible plan refinements
15:            for φ*_i ∈ Φ* do
16:                // compute flaw's solutions
17:                N_φ*_i = {n_1, ..., n_t} ← HandleFlaw(φ*_i, π)
18:                // check if the current flaw can be solved
19:                if N_φ*_i = ∅ then
20:                    Backtrack(π, Dequeue(F_pc))
21:                end if
22:                for n_j ∈ N_φ*_i do
23:                    // expand the search space
24:                    F_pc ← Enqueue(n_j, S)
25:                end for
26:            end for
27:        else
28:            // non pseudo-controllable plan
29:            F_¬pc ← Enqueue(makeNode(π), S)
30:        end if
31:        // check the fringe of the search space
32:        if IsEmpty(F_pc) ∧ ¬IsEmpty(F_¬pc) then
33:            // try to find a non pseudo-controllable solution
34:            π ← Refine(π, Dequeue(F_¬pc))
35:        else if ¬IsEmpty(F_pc) then
36:            // go on looking for a pseudo-controllable plan
37:            π ← Refine(π, Dequeue(F_pc))
38:        else
39:            return Failure
40:        end if
41:    end while
42:    // get solution plan
43:    return π
44: end function
```

**Hierarchy-based Flaw Selection Heuristcs.** The behavior of the planning procedure shown in Algorithm 1 is determined by the particular search strategy $\mathcal{S}$ and the flaw selection heuristic $\mathcal{H}$. Specifically, flaw selection can strongly affect the performance of

the planning process even if it does not represent a backtracking point of the algorithm. Indeed, each solution of a flaw determines a branch of the search tree. A flaw selection heuristic is supposed to encapsulate smart criteria for suitably evaluating flaws during planning. A good selection of the next flaw to solve can *prune* the search space by cutting off branches that would lead to *unnecessary* or *redundant* refinements of the plan. In addition, leveraging the hierarchical modeling approach presented in Section 2.2, a suitable heuristics to guide the selection of flaws can be defined by means of the domain knowledge. The work [20] has shown that it is possible to define a hierarchy-based heuristic capable of leveraging such information and improve the planning capabilities of timeline-based applications.

**Algorithm 2** The hierarchy-based flaw selection heuristic

```
1: function SELECTFLAWS(π)
2:     // initialize the set of flaws
3:     Φ ← ∅
4:     // extract the hierarchy of the domain
5:     H_π = {h_1, ..., h_m} ← extractHierarchy (π)
6:     for h_i = {sv_{i,1}, ..., sv_{i,k}} ∈ H_π do
7:         if Φ = ∅ then
8:             // detect flaws on state variables composing the hierarchical level h_i
9:             for sv_{i,j} ∈ h_i = {sv_{i,1}, ..., sv_{i,k}} do
10:                // select detected flaws
11:                Φ ← detctFlaws (sv_{i,j})
12:            end for
13:        end if
14:    end for
15:    // get selected flawsts
16:    return Φ
17: end function
```

Algorithm 2 depicts the *SelectFlaws* procedure in Algorithm 1 (row 14) according to such hierarchy-based heuristic. The heuristic takes into account the hierarchical structure of the domain and select flaws that belong to the most independent state variables of the domain (i.e., flaws concerning state variables that come first in the hierarchy). The rationale behind the heuristic is that the hierarchical structure encapsulates dependencies among the state variables composing a planning domain. Thus, the resolution of flaws concerning state variables at the *higher* levels of the hierarchy (i.e., the most independent variables) can simplify the resolution of flaws concerning state variables at the *lower* levels of the hierarchy (i.e., the most dependent variables).

### 3.2 Timeline-based Plan Execution

The most innovative aspect in PLATINUM is its ability to perform also plan execution relying on the same semantics of timelines in the pursued planning approach [6]. Therefore, PLATINUM executives leverage information about *temporal uncertainty* in order to properly manage and adapt the execution of plans. In general, the execution of a plan is a complex process which can fail even if a plan is valid with respect to the domain specification. During execution, the system must *interact* with the environment,

which is *uncontrollable*. Such dynamics can affect or even prevent the correct execution of plans. A *robust* executive system must cope with such exogenous events and dynamically *adapt* the plan accordingly during execution.

**Controllability-Aware Execution.** The execution process consists of *control cycles* whose frequency determines advancement of time and the discretization of the temporal axis in a number of units called *ticks*. Each *control cycle* is associated with a *tick* and realizes the execution procedure. Broadly speaking, the execution procedure is responsible for detecting the actual behavior of the system (*closed-loop* architecture), for verifying whether the system and also the environment behave as expected from the plan and for starting the execution of the activities of the plan.

**Algorithm 3** The PLATINUM executive control procedure

```
1: function EXECUTE(Π, C)
2:     // initialize executive plan database
3:     π_exec ← Setup(Π)
4:     // check if execution is complete
5:     while ¬CanEndExecution(π_exec) do
6:         // wait a clock's signal
7:         τ ← WaitTick(C)
8:         // handle synchronization phase
9:         Synchronize(τ, π_exec)
10:        // handle dispatching phase
11:        Dispatch(τ, π_exec)
12:    end while
13: end function
```

Algorithm 3 shows the pseudo-code of the general PLATINUM executive procedure. The procedure is composed by two distinct phases, the *synchronization phase* and the *dispatching phase*. At each tick (i.e., control cycle) the synchronization phase manages the received execution feedbacks/signals in order to build the current status of the system and the environment. If the current status is valid with respect to the plan, then the dispatching phase *decides* the next activities to be executed. Otherwise, if the current status does not fit the plan, an *execution failure* is detected and *replanning* is needed. In such a case, the current plan does not represent the actual status of the system and the environment and therefore replanning allows the executive to continue the execution process with a new plan, which has been generated according to the *observed* status and the executed part of the *original plan*.

Algorithm 4 shows the pseudo-code of the synchronization procedure of the executive. The synchronization phase monitors the execution of the plan by determining whether the system and the environment are aligned with respect to the expected plan. Namely, at each iteration the synchronization phase builds the current situation by taking into account the current execution time, the expected plan and the feedbacks received during execution. A *monitor* is responsible for propagating observations concerning the actual duration of the dispatched activities and detecting discrepancies between the real world and the plan. The executive receives feedbacks about the successful execution of dispatched commands or failure. The monitor manages these feedbacks in

order to detect if the actual duration of tokens comply with the plan. If the feedbacks comply with the plan, then the execution of the plan can proceed. Otherwise, a *failure* is detected because the current situation does not fit the expected plan and the executive reacts accordingly (*replanning*).

**Algorithm 4** The PLATINUM executive procedure for the synchronization phase

```
1: function SYNCHRONIZE(τ, π_exec)
2:     // manage observations
3:     O = {o_1, ..., o_n} ← GetObservations(π_exec)
4:     for o_i ∈ O do
5:         // propagate the observed end time
6:         π_exec ← PropagateObservation(τ, o_i)
7:     end for
8:     // check if observations are consistent with the current plan
9:     if ¬IsConsistent(π_exec) then
10:        // execution failure
11:        return Failure
12:    end if
13:    // manage controllable activities
14:    A = {a_i, ..., a_m} ← GetControllableActivities(π_exec)
15:    for a_i ∈ A do
16:        // check if activity can end execution
17:        if CanEndExecution(τ, a_i, π_exec) then
18:            // propagate the decided end time
19:            π_exec ← PropagateEndActivity(τ, a_i)
20:        end if
21:    end for
22: end function
```

**Algorithm 5** The PLATINUM executive procedure for the dispatching phase

```
1: function DISPATCH(τ, π_exec)
2:     // manage the start of (all) plan's activities
3:     A = {a_i, ..., a_m} ← GetActivities(π_exec)
4:     for a_i ∈ A do
5:         // check if activity can start execution
6:         if CanStartExecution(τ, a_i, π_exec) then
7:             // propagate the decided start time
8:             π_exec ← PropagateStartActivity(τ, a_i)
9:             // actually dispatch the related command to the robot
10:            SendCommand(a_i)
11:        end if
12:    end for
13: end function
```

Algorithm 5 shows the pseudo-code of the dispatching procedure of the executive. The dispatching phase manages the actual execution of the plan. Given the current situation and the current execution time, the dispatching step analyzes the plan $\pi_{exec}$ in order to find the tokens that can start execution and dispatches the related commands to the underlying system. Namely, the dispatching step allows the executive to advance execution and decide the next tokens to execute. Thus, a *dispatcher* is responsible for making dispatching decisions of plan's tokens. For each token, the dispatcher checks the related *start condition* by analyzing the token's scheduled time and any dependency

with other tokens of the plan. If the start condition holds, then the dispatcher can decide to start executing the token (i.e., the dispatcher propagates the scheduled start time into the plan).

### 3.3 Token Lifecycle

A plan and its temporal relations encapsulate a set of execution dependencies that must be taken into account when executing timelines. Besides the scheduled temporal bounds, such dependencies specify whether the executive can actually start or end the execution of a token. Let us consider for example a plan where the temporal relation *A before B* holds between tokens *A* and *B*. Such a temporal relation encapsulates an execution dependency between token A and token B. The executive can start the execution of token B if and only if the execution of token A is over. In addition, the executive must take into account *controllability properties* of tokens. Different controllability properties entail different execution policies of tokens and therefore different *lifecycles*.

*Controllable tokens* are completely under the executive control. In this case the executive can decide both the start time and the duration of the execution of this type of tokens. Both decisions are controllable. *Partially-controllable tokens* are not completely under the control of the executive. Tokens of such a type are under the control of the *environment*. The executive can decide the start time (i.e., the dispatching time) while it can only *observe* the actual execution and update the plan according to the *execution feedbacks* received from the environment. Finally, *uncontrollable tokens* are completely outside the control of the executive. The executive can neither decide the start nor the end of the execution. Both "events" are under the control of the environment. *Execution feedbacks* concern both the start time and the end time of the execution and therefore the executive can only update/adapt the controllable part of the plan accordingly.

## 4 Deployment in a Real Scenario

A separate work [16] describes how an instance of PLATINUM has been deployed in a manufacturing case study integrating the task planning technology described above with a motion planning system for industrial robots [15]. In that integration, PLATINUM and its features are leveraged to implement an integrated task and motion planning system capable of selecting different *execution modalities* for robot tasks according to the expected *collaboration* of the robot with a human operator. This is the result of a tight integration of PLATINUM with a motion planning system. Indeed, the pursued approach realizes an *offline analysis* of the production scenarios in order to synthesize a number of collision-free *robot motion trajectories* for each collaborative task with different *safety* levels. Each trajectory is then associated with an expected temporal execution bound and represents a tradeoff between "speed" of the motion and "safety" of the human. The integrated system has been deployed and tested in laboratory on an assembly case study similar to collaborative assembly/disassembly scenario described above. In [16], an empirical evaluation is provided in order to assess the overall productivity of the HRC cell while increasing the involvement of the robots (i.e., increasing

the number of tasks the robot is allowed to perform). The results show the effectiveness of PLATINUM in finding well suited distribution of tasks between the human and the robot in different scenarios with an increasing workload for the control system. Specifically, the PLATINUM instance results as capable of increasing the productivity of the production process without affecting the safety of the operator.

Before concluding the paper it is worth underscoring that for lack of space this paper does not concern an experimental evaluation of PLATINUM features. Some focalized experiments are contained in [16] while a wider experimental campaign is undergoing and will constitute an important pillar for a future longer report.

## 5 Conclusions

In this paper, a recent evolution of a timeline-based planning framework has been presented. In particular, taking also advantage of the needs coming from the FOURBYTHREE project, the EPSL planner has been endowed of novel features for structured modeling, synthesizing plans under uncertainty and plan execution functionality. The obtained framework, called PLATINUM, is currently supporting the need of adaptability in new domains. The table below summarizes the main features and capability introduced in PLATINUM and points out differences with respect to EPSL.

| | EPSL | PLATINUM |
|---|---|---|
| Representation | Temporal Flexibility | Temporal Flexibility with Uncontrollability |
| Solving | Hierarchical | Hierarchical with Uncertainty |
| Execution | Not Supported | Plan Execution with Uncertainty |

PLATINUM represents a uniform framework for planning and execution with timelines under uncertainty. It relies on a well-defined formalization of the timeline-based approach [6] and has proven to be particularly suited for addressing HRC scenarios. It is worth underscoring how PLATINUM enters in the current state of the art in a sub-area of planning systems for robotics together with CHIMP [18], HATP [9], meta-CSP planner [11], FAPE [2] that creates a *current generation* of planners for robotics whose goal is to evolve with respect to classical temporal frameworks such as, for instance, EUROPA [1] and IXTET [8]. Finally, leveraging the work in [14], a current research effort is related to the extension of the knowledge engineering framework for PLATINUM to enable use by *non specialist* planning users. Our goal, as already said, is to enable the use of our technology in different industrial settings. To this aim, creating a tool for non specialists for modelling, configuring and implementing plan-based controllers is an important goal to pursue.

## References

1. Barreiro, J., Boyce, M., Do, M., Frank, J., Iatauro, M., Kichkaylo, T., Morris, P., Ong, J., Remolina, E., Smith, T., Smith, D.: 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)

2. Bit-Monnot, A.: Temporal and Hierarchical Models for Planning and Acting in Robotics. Ph.D. thesis, Doctorat de l'Université Federale Toulouse Midi-Pyrenees (2016)
3. Cesta, A., Finzi, A., Fratini, S., Orlandini, A., Tronci, E.: Validation and Verification Issues in a Timeline-Based Planning System. Knowledge Engineering Review 25(3), 299–318 (2010)
4. Cesta, A., Orlandini, A., Bernardi, G., Umbrico, A.: Towards a planning-based framework for symbiotic human-robot collaboration. In: 21th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). IEEE (2016)
5. Cialdea Mayer, M., Orlandini, A.: An executable semantics of flexible plans in terms of timed game automata. In: The 22nd International Symposium on Temporal Representation and Reasoning (TIME). IEEE (2015)
6. Cialdea Mayer, M., Orlandini, A., Umbrico, A.: Planning and execution with flexible timelines: a formal account. Acta Informatica 53(6-8), 649–680 (2016)
7. Freitag, M., Hildebrandt, T.: Automatic design of scheduling rules for complex manufacturing systems by multi-objective simulation-based optimization. {CIRP} Annals - Manufacturing Technology 65(1), 433 – 436 (2016)
8. Ghallab, M., Laruelle, H.: Representation and control in ixtet, a temporal planner. In: 2nd Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS). pp. 61–67 (1994)
9. Lallement, R., de Silva, L., Alami, R.: HATP: an HTN planner for robotics. CoRR abs/1405.5345 (2014), http://arxiv.org/abs/1405.5345
10. Lemai, S., Ingrand, F.: Interleaving Temporal Planning and Execution in Robotics Domains. In: AAAI-04. pp. 617–622 (2004)
11. Mansouri, M., Pecora, F.: More knowledge on the table: Planning with space, time and resources for robots. In: 2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014. pp. 647–654. IEEE (2014)
12. Maurtua, I., Pedrocchi, N., Orlandini, A., Fernández, J.d.G., Vogel, C., Geenen, A., Althoefer, K., Shafti, A.: Fourbythree: Imagine humans and robots working hand in hand. In: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA). pp. 1–8 (Sept 2016)
13. Morris, P.H., Muscettola, N., Vidal, T.: Dynamic Control of Plans With Temporal Uncertainty. In: International Joint Conference on Artificial Intelligence (IJCAI). pp. 494–502 (2001)
14. Orlandini, A., Bernardi, G., Cesta, A., Finzi, A.: Planning meets verification and validation in a knowledge engineering environment. Intelligenza Artificiale 8(1), 87–100 (2014)
15. Pellegrinelli, S., Moro, F.L., Pedrocchi, N., Tosatti, L.M., Tolio, T.: A probabilistic approach to workspace sharing for human–robot cooperation in assembly tasks. {CIRP} Annals - Manufacturing Technology 65(1), 57 – 60 (2016)
16. Pellegrinelli, S., Orlandini, A., Pedrocchi, N., Umbrico, A., Tolio, T.: Motion planning and scheduling for human and industrial-robot collaboration. {CIRP} Annals - Manufacturing Technology pp. – (2017)
17. Py, F., Rajan, K., McGann, C.: A systematic agent framework for situated autonomous systems. In: AAMAS. pp. 583–590 (2010)
18. Stock, S., Mansouri, M., Pecora, F., Hertzberg, J.: Online task merging with a hierarchical hybrid task planner for mobile service robots. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. pp. 6459–6464 (Sept 2015)
19. Umbrico, A., Cesta, A., Cialdea Mayer, M., Orlandini, A.: Steps in Assessing a Timeline-Based Planner, pp. 508–522. Springer International Publishing (2016)
20. Umbrico, A., Orlandini, A., Cialdea Mayer, M.: Enriching a temporal planner with resources and a hierarchy-based heuristic. In: AI*IA 2015, Advances in Artificial Intelligence, pp. 410–423. Springer International Publishing (2015)
21. Vidal, T., Fargier, H.: Handling Contingency in Temporal Constraint Networks: From Consistency To Controllabilities. JETAI 11(1), 23–45 (1999)