Design of Timeline-based Planning Systems for Safe Human-Robot Collaboration

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

[1] Institute of Cognitive Science and Technology,
CNR – National Research Council of Italy, Rome
Email: {name.surname}@istc.cnr.it
[2] Department of Engineering, University "Roma TRE"
Rome (RM), Italy
Email: cialdea@ing.uniroma3.it

Abstract. During the last decade, industrial collaborative robots have entered assembly cells supporting human workers in repetitive and physical demanding operations. Such Human-Robot Collaboration (HRC) scenarios entail many open issues. The deployment of highly flexible and adaptive plan-based controllers capable of preserving productivity while enforcing human safety is then a crucial requirement. The deployment of plan-based solutions entail knowledge engineers and roboticists interactions in order to design well suited models of robotic cells considering both operational and safety requirements. So, the ability of supporting knowledge engineering for integrating high level and low level control (also from non specialist users) can facilitate deployment of effective and safe solutions in different industrial settings. In this chapter, we will provide an overview of some recent results concerning the development of a task planning and execution technology and its integration with a state of the art Knowledge Engineering environment to deploy safe and effective solutions in realistic manufacturing HRC scenarios. We will briefly present and discuss a HRC use case to demonstrate the effectiveness of such integration discussing its advantages.

1 Introduction

During the last decade, *industrial robotic* systems have entered assembly cells supporting human workers in repetitive and physical demanding operations. The co-presence of robots and humans in a shared environment entails many issues to be properly addressed requiring *robust controllers* capable of preserving *productivity* and enforcing *human safety* [1]. Namely, Human-Robot Collaboration (HRC) scenarios entail challenges on *physical interactions*, to always guarantee *safety* of human operators, and activities *coordination*, to improve cell productivity. Thus, the deployment of highly flexible and adaptive controllers capable of preserving productivity while enforcing human safety is a crucial requirement.

Flexible plan-based solutions are a key enabling feature of HRC controllers where robot motions must be continuously adapted to the presence of humans, which act as uncontrollable "agents" in the environment. Their presence entails the ability of evaluating robot execution time variability and, in this sense, standard control methods are

not fully effective. Moreover, the integration of Planning and Scheduling (P&S) technology with Knowledge Engineering solutions and, more specifically, with Verification and (V&V) techniques is a key element to synthesize safety critical systems in robotics [2]. Indeed, the deployment of plan-based solutions requires domain experts (i.e., production engineers), knowledge engineers and roboticists to deeply interact in order to design well suited models of robotic cells considering both operational and safety requirements [3]. Therefore, the ability of supporting knowledge engineering for integrating high level and low level control (also from non specialist users) can facilitate deployment of effective and safe solutions in different industrial settings.

Since a decade, a research initiative has been started to investigate the possible integration of a timeline-based planning framework (APSI-TRF [4]) and V&V techniques based on Timed Game Automata (TGA) [5] to automatically synthesize a robot controller that guarantees robustness and safety properties [6, 7]. Indeed, some plan-based controllers rely on temporal planning mechanisms capable of dealing with coordinated task actions and temporal flexibility (e.g., [8, 9]) that leverage temporal planners (e.g., [10, 11]). Unfortunately, these systems do not allow an explicit representation of *uncontrollability* features. Consequently, the resulting controllers are not endowed with the *robustness* needed to deal with the *temporal uncertainty* of HRC scenarios and controllability issues [12]. These system usually rely on *replanning mechanisms* that may however strongly penalize the production performance. The long-term research goal is to realize a robust task planning system enabling flexible, safe and efficient HRC. In [13], the general pursued approach is presented aiming at realizing controllers capable to dynamically coordinate tasks according to the behaviors of human workers.

This chapter provides an overview of the results collected in the last decade concerning the development of a task planning and execution framework (PLATINUM) [14, 15] and its integration with a Knowledge Engineering ENvironment (KEEN) describing its deployment in safe and effective solutions for manufacturing HRC scenarios. KEEN [16] is a knowledge engineering environment with Verification and Validation features based on Timed Game Automata model checking. PLATINUM is a timeline-based planning systems capable of supporting flexible temporal planning and execution with uncertainty. Also, more recent results on the development of a task planning and execution technology deployed in realistic manufacturing scenarios will be presented. Specifically, the paper presents an Engineering & Control Environment which integrates a task planning system with an *engineering environment* taylored to support robust human-robot collaboration.

2 Fostering Autonomy via Timeline-based Planning and Execution

Planning for real world problems with explicit temporal constraints is a challenging problem. Among different approaches, the use of flexible timelines in Planning and Scheduling (P&S) has been shown to be successful in a number of concrete applications, such as, for instance, autonomous space systems [17, 18, 19]. Timeline-based planning has been introduced by Muscettola [19], under a modeling assumption inspired by classical control theory. A planning problem is modeled by identifying a set of relevant components whose temporal evolution must be controlled to obtain a desired

behavior. Components represent logical or physical subsystems whose state may vary over time. The behavior of the domain features under control are modeled as temporal functions whose values have to be decided over a temporal horizon. Such functions are synthesized during problem solving by posting planning decisions. The evolution of a single temporal feature over a temporal horizon is called the *timeline* of that feature.

In general, plans synthesized by temporal P&S systems may be (i) temporally flexible and (ii) not fully controllable. Time flexibility reflects on modeling plans as made up of *flexible* timelines, describing transition events that are associated with temporal intervals (with given lower and upper bounds), instead of exact temporal occurrences. In other words, a flexible plan describes an envelope of possible solutions with the aim of facing uncertainty during actual execution. As a matter of fact, many P&S architectures return flexible plans, which are commonly accepted to be less brittle than fully specified plans, when coping with execution. The second above-mentioned property is due to the fact that not every value transition in a plan is under the system control, as events exist that depend on the *environment*. The execution of a flexible plan is usually under the responsibility of an executive system that forces value transitions over the timelines dispatching commands to the concrete system, while continuously accepting feedback and, thus, monitoring plan execution. In such cases, the execution time of controllable tasks should be chosen so that they can face uncontrollable events. This is known as the *controllability problem* [20].

2.1 A Theoretical Framework

After several several attempts, a formal framework has been presented to provide a unique theoretical background on timelines [21]. This section provides a brief informal overview of the basic notions regarding flexible timelines and plans.

A timeline-based planning domain contains the characterization of a set of *state variables*, representing the components of a system. A *state variable* x is characterized by the set of values it may assume, denoted by $\mathrm{values}(x)$, possible upper and lower bounds on the duration of each value, and rules governing the correct sequencing of such values. A *timeline* for a state variable is made up of a finite sequence of valued intervals, called *tokens*, each of which represents a time slot where the variable assumes a given value. In general, timelines may be *flexible*, i.e., the start and end times of each of its tokens are not necessarily fixed time points, but may range in given intervals. For the sake of generality, temporal instants and durations are taken from an infinite set of non negative numbers $\mathbb{T}$, including 0. The notation $\mathbb{T}^\infty$ will be used to denote $\mathbb{T} \cup \{\infty\}$, where $t < \infty$ for every $t \in \mathbb{T}$.

Tokens in a timeline for the state variable x are denoted by expressions of the form x^i, where the superscript indicates the position of the token in the timeline. Each token x^i is characterized by a value $v_i \in \mathrm{values}(x)$, an end time interval $[e_i, e'_i]$ referred to as $\mathrm{end\_time}(x^i)$, and a duration interval $[d_i, d'_i]$ (as usual, the notation $[x, y]$ denotes the closed interval $\{t \mid x \leq t \leq y\}$). The start time interval $\mathrm{start\_time}(x^i)$ of the token x^i is $[0, 0]$ if x^i is the first token of the timeline (i.e. $i = 1$), otherwise, if $i > 1$, $\mathrm{start\_time}(x^i) = \mathrm{end\_time}(x^{i-1})$. So, a token has the form $x^i = (v_i, [e_i, e'_i], [d_i, d'_i])$ and a timeline is a finite sequence of tokens $x^1, \ldots, x^k$. The metasymbol FTL (FTL_x) will henceforth be used to denote a timeline (for the state variable x), and **FTL** to

denote a set of timelines. Being tokens flexible, their exact start end end times will be decided at execution time. Tokens can be either *controllable* (the controller can decide both their start and end time), or *uncontrollable* (both start and end time depend on the environment's choices), or *partially controllable* (the controller can decide when to start them , but their exact duration is outside the system's control). Each token is consequently equipped also with a *controllability tag*, identifying the class it belongs to.

A *scheduled timeline* is a particular case where each token has a singleton $[t, t]$ as its end time, i.e., the end times are all fixed. A *schedule* of a timeline FTL_x is essentially obtained from FTL_x by narrowing down token end times to singletons (time points) in such a way that the duration requirements are fulfilled. In a given timeline-based domain, the behavior of state variables may be restricted by requiring that time intervals with given state variable values satisfy some temporal constraints. Such constraints are stated as a set of *synchronization rules* which relate tokens on possibly different timelines through temporal relations between intervals or between an interval and a time point. These temporal relations refer to token start or end points, that will henceforth be called *events*. If **FTL** is a set of timelines and $\mathrm{tokens}(\mathbf{FTL})$ the set of the tokens in **FTL**, then the set $\Upsilon(\mathbf{FTL})$ of the *events* in **FTL** is the set containing all the expressions of the form $\mathrm{start\_time}(x^i)$ and $\mathrm{end\_time}(x^i)$ for $x^i \in \mathrm{tokens}(\mathbf{FTL})$. A *temporal relation* on tokens has then one of the following forms:

$$p \leq_{[lb,ub]} p' \quad p \leq_{[lb,ub]} t \quad t \leq_{[lb,ub]} p$$

where $p, p' \in \Upsilon(\mathbf{FTL})$, $t, lb \in \mathbb{T}$ and $ub \in \mathbb{T}^\infty$.

Intuitively, $p \leq_{[lb,ub]} p'$ states that the token start/end point denoted by p occurs from lb to ub time units before that denoted by p'; $p \leq_{[lb,ub]} t$ states that the token start/end point denoted by p occurs from lb to ub time units before the time point t and the third relation that it occurs from lb to ub time units after t. Other relations between tokens ([22]) can be defined in terms of the primitive ones, e.g.: $x^i \,\mathrm{before}_{[lb,ub]}\, y^j$ is the same as $\mathrm{end\_time}(x^i) \leq_{[lb,ub]} \mathrm{start\_time}(y^j)$; $x^i \,\mathrm{during}_{[lb_1,ub_1][lb_2,ub_2]}\, y^j$ can be defined as $\mathrm{start\_time}(y^j) \leq_{[lb_1,ub_1]} \mathrm{start\_time}(x^i)$ and $\mathrm{end\_time}(\mathrm{x^i}) \leq_{[lb_2,ub_2]} \mathrm{end\_time}(y^j)$; a *contains* relation is its converse: $x^i \,\mathrm{contains}_{[lb_1,ub_1][lb_2,ub_2]}\, y^j$ if and only if $y^j \,\mathrm{during}_{[lb_1,ub_1][lb_2,ub_2]}\, x^i$.

Temporal relations are also used to state the synchronization rules of the planning domain. Here, it is sufficient to say that such rules allow to state requirements of the following form: for every token x_0^i where the state variable x_0 assumes the value v_0, there exist tokens $x_1^{i_1}, \ldots, x_n^{i_n}$ where the state variables $x_1, \ldots, x_n$ hold some given specified values, and all these tokens are related one to another by some given temporal relations. Unconditioned synchronization rules are also allowed, and are useful for stating both domain invariants and planning goals.

A *flexible plan* Π is a pair $(\mathbf{FTL}, \mathcal{R})$, where **FTL** is a set of timelines and $\mathcal{R}$ is a set of temporal relations, involving tokens in some timelines in **FTL**. An *instance* of the flexible plan $\Pi = (\mathbf{FTL}, \mathcal{R})$, is any schedule of **FTL** that satisfies every relation in $\mathcal{R}$. In order for a flexible plan $\Pi = (\mathbf{FTL}, \mathcal{R})$ to satisfy a synchronization rule it must be the case that $\mathcal{R}$ contains temporal relations guaranteeing what the rule requires. For the formal definitions the reader is again referred to [21], where it is also proved that

whenever a flexible plan satisfies (in this sense) all the syncronization rules of a domain, then also any of its instances does.

2.2 PLATINUm: a Timeline-based Planning and Acting Framework

Born as a followup of APSI-TRF, PLATINUM[3] is a general-purpose timeline-based planning and execution framework capable of dealing with *temporal uncertainty* and *controllability issues* [14, 23] that complies with the formalization given in section 2.1. PLATINUM is able to deal with *uncontrollable dynamics* at both planning and execution time. Its solving process pursues a *plan refinement approach* which consists in iteratively refining a partial plan by reasoning in terms of *flaws* that must be solved. Flaw selection is supported by dedicated heuristics that guide the planning procedure. A PLATINUM-based planner relies on a set of data structures and algorithms called respectively *components* and *resolvers*. Components model the types of features that may compose a planning domain. They specify the set of states and constraints that characterize the temporal behaviors of a particular type of domain feature. Resolvers are dedicated algorithms that encapsulate the logic for building valid temporal behaviors of a particular component. The reader may refer to [14, 23] for a more detailed description of the framework and the solving approach. However, it is important to point out that resolvers are not responsible for making decisions during the search process. They are responsible for detecting flaws on a component and computing all possible solutions of such flaws in order to guarantee completeness of the search. Each solution of a flaw represents a branch in the search and it is up to the planner deciding which flaw to solve and which solution to apply for search expansion (i.e., plan refinement). The types of flaws a PLATINUM-based planner is capable to deal with depend on the set of components and resolvers available in the framework. PLATINUM provides *state variables components* and the related resolvers that allow a planner to build valid *timelines* according to the semantics proposed in [21]. Thus, PLATINUM has been extended by adding new components and new resolvers in order to properly deal with *discrete* and *reservoir* resources [15].

3 KEEN: Knowledge Engineering ENvironment

In order to foster the deployment of reliable timeline-based P&S applications, the development of a Knowledge Engineering ENvironment (KEEN) for Timeline-based planning has been pursued [16]. Here, the context in which the needs for a new tool emerged is described, and the main requirements for the new environment are presented. Then, the core design choices that lead the development of KEEN are discussed, divided accordingly to the features that they are intended to support.

KEEN is an Open Source software released under the Eclipse Public License, version 1.0, and as such its source code can be downloaded from its GitHub repository[4].

[3] https://github.com/pstlab/PLATINUm

[4] `https://github.com/ugilio/keen`
The reader interested in knowing how to use KEEN can find a detailed description at `https://ugilio.github.io/keen/userguide`.

There were no tools for Timeline-based Planning supporting graphical modeling of the solution, and neither domain validation and plan verification; this alone might justify the creation of a new tool to specifically fill this niche. However, the motivations that led to the development of KEEN were due to some specific needs that arose in the Planning and Scheduling Technology Laboratory at the National Research Council of Italy (CNR-ISTC). Specifically, some research done in the field of Verification and Validation (V&V) for Timeline-based Planning [2, 6, 24]: specifically, a formalism has been developed to translate planning domains in the form of Timed Game Automata (TGA) [57], so as to make it possible to employ existing model checkers to verify the translated domain. Regarding plan verification, other work was done [28, 26] to also encode flexible plans generated by a timeline-based planner in the form of Timed Game Automata, thus making it possible to verify generated plans in the same way as mentioned earlier. Additionally, at the PST there was the need to more efficiently exploit the technologies underlying the planners and frameworks that have been developed there during the years; a common language to express domains and problems, DDL [38], did exist, but there were no integrated tools to ease the development of timeline-based systems; this situation was far less than ideal because: the work practice required to write domains using standard text editors, then manually invoking a planner on the files, and possibly execute tools to translate domains and plans to TGA encoding and running other tools to verify them; this was cumbersome and time-consuming in the first place; maintaining systems or refining them after some time had passed was a daunting task; if this development style was already very complicated for planning experts, it was a real stopper for beginners who approached the field for the first time.

This suggested that a tool to make developing comfortable was an absolute minimum; moreover, given the availability of Knowledge Engineering tools with support for graphical modeling in other fields of Automated Planning and Scheduling [74, 63, 75], it would have been desirable to have one for Timeline-based planning too. And finally, a truly integrated environment had to also incorporate the work done on validation and verification to make it accessible to the developer in a simple and productive form.

3.1 Knowledge Engineering and Verification & Validation Features in KEEN

Requirements for the new environment were elicited by interviewing the potential new users of the system: developers of planning domains that already have a deep knowledge of timeline-based planning, and their colleagues which were marginally interested in writing code in detail, but were more concerned about the ability of sketching domains to be further refined by planning experts, or which needed to visualize the high-level information about the work done by them. A list of broad, high-level requirements has been identified as the following: Provide all the traditional features of a modern Integrated Development Environment: (a) Syntax highlighting; (b) Code assist; (c) Tree-view of syntactic elements; (d) Error detection. Leverage existing, well-known tools already in use at PST. Support graphical editing of domains. Support Round-Trip Engineering: (a) be able to edit both the textual and graphical representations in a synchronized way; (b) without having to import and export code to/from the graphical representation; (c) synchronization between the views should be automatic. Make it possible

to use existing planners from the environment. Support code sharing using popular version control systems. Enable users to easily validate domains and verify plans. These requirements all refer to the quite common and well-known field of Integrated Development Environments, which every developer is familiar with, given that these tools are commonly used in their daily work; for these reason, it has not been necessary to conduct in-depth analyses to better explore the domain of application of some requirements: instead, it has been enough clarify some aspects via informal conversations and releasing often early versions of the product to have a continuous feedback on the work being done.

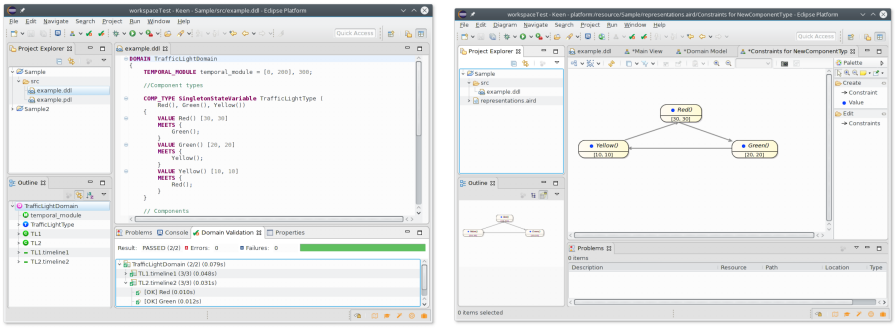

Fig. 1: KEEN graphical interface: eclipse-generated text highliths (left-side); a graphical view of the planning domain (right-side).

KEEN, built around APSI-TRF and now applied to PLATINUM, was designed as a set of active services to support knowledge engineering. We distinguish between two different service layers, i.e., a set of *Knowledge Engineering Services* (upper part of Fig. 2) and a set of V&V *services* (lower part of Fig. 2). The *Knowledge Engineering Services* provide "classical" KE functionalities specifically developed for timeline-based planning. At present, this part is composed of (1) a *Domain/Problem Editing and Visualization* module that support synthesis and modification of planning models, and (2) a *Plan Editing and Visualization* module that helps inspection, analysis, and direct manipulation of solution plans. The V&V *Services* contribute to the KEEN tool with a set of fully automated V&V features obtained making operational some research results presented in [25, 26, 7]. The V&V functionalities are all based on model checking for Timed Game Automata (TGA) and rely on UPPAAL-TIGA [27] as the verification engine. UPPAAL-TIGA extends UPPAAL [28] by providing a toolbox for the specification, simulation, and verification of real-time games. Somehow such model checker constitutes an additional core engine for KEEN. The general KEEN concept is depicted in Fig. 2.

A *TGA Encoding* module provides the basic TGA automatic translation for P&S specification [25] which constitutes the basis for implementing the KEEN V&V services: (1) the *Domain Validation* supports the model building activity allowing to check the P&S model with respect to system requirements; (2) the *Planner Validation* assesses the P&S solver with respect to a given set of requirements. In this regard, two

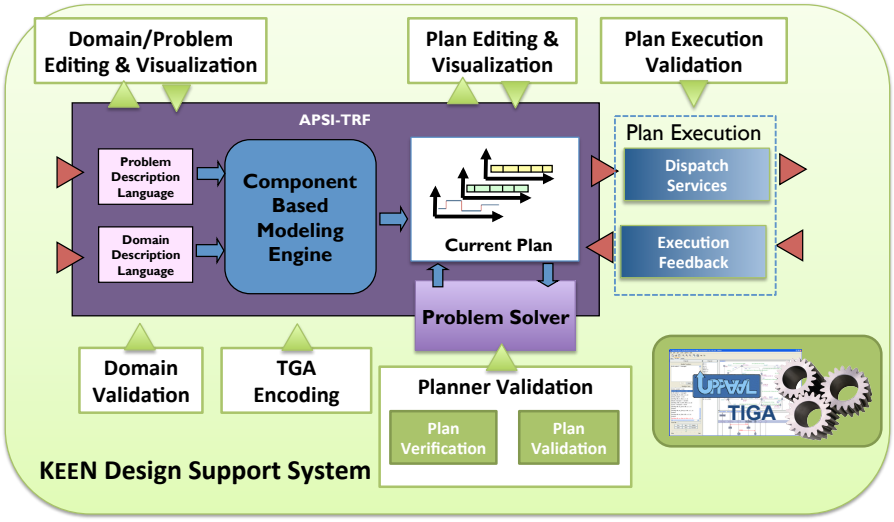

Fig. 2: The general V&V architecture in KEEN.

sub-modules are further deployed, i.e., *Plan Verification* to verify the correctness of the solution plans and *Plan Validation* to evaluate their adequacy; finally, (3) a TGA-based approach to *Plan Controller Synthesis* [7] is able to enforce robust execution of solution plans through the generation of robust plan controllers.

The pursued idea is the integration of KEEN with either an accurate simulator of a real environment or a real physical system (e.g., a robot). In this context, it is possible to take advantage of all KEEN functionalities during the different design phases, i.e., from initial design to actual solution execution and continuously exploiting the combination of "classical" KE and V&V functionalities. Users can also ask for solution plan generation by means of the KEEN functionalities. Indeed, exploiting the PLATINUM capabilities, this may be performed by means of the specific planners. As for the domain, the plan representation is completely handled by PLATINUM and a specific language generation component is deputed to the generation of a source file encoded with a *Problem Description Language* syntax. The user can then modify the generated solution plan and ask KEEN to perform V&V tasks.

4 Deploying Task Planning solutions for Safe Human-Robot Collaboration

In order to deploy reliable task planning solutions for safe Human-Robot Collaboration, we may consider a collaborative robotic workcell as a bounded connected space with two agents located in it, a human and a robot, and their associated equipment [29]. The robot system in a workcell consists of a robotic arm with its tools, its base and possibly additional support equipment. In such workcell, four different degrees of interaction between a human operator and the robot can be defined [30] in which they may need to occupy the same spatial location and interact according to different modalities: *Inde-*

pendent, the human and the robot operate on separate workpieces without collaboration, i.e., independently from each other; *Synchronous*, the human and the robot operate on sequential components of the same workpiece, i.e., one can start a task only after the other has completed a preceding task; *Simultaneous*, the human and the robot operate on separate tasks on the same workpieces at the same time; *Supportive*, the human and the robot work cooperatively to complete the processing of a single workpiece, i.e., they work simultaneously on the same task. Different interaction modalities entails the robot to be endowed with different safety settings while executing tasks.

4.1 A Specific Human-Robot Collaboration Case Study

In Manufacturing, different production processes can be performed with HRC solutions, i.e., assembly/disassembly of parts, welding operations, large parts management, machine tending, etc. Among these, here we describe a specific case study considered in a research project named FourByThree [5] [31]. This case study corresponds to a real production industry with different relevant features for our perspective (e.g., space sharing, collaboration; interaction needs, etc). The overall production process consists of a metal die which is used to produce a wax pattern in a injection machine. Once injected, the pattern is taken out the die. Several patterns are assembled to create a cluster. The wax assembly is covered with a refractory element, creating a shell (this process is called investing). The wax pattern material is removed by the thermal or chemical means. The mould is heated to a high temperature to eliminate any residual wax and to induce chemical and physical changes in the refractory cover. The metal is poured into the refractory mould. Once the mould has cooled down sufficiently, the refractory material is removed by impact, vibration, and high pressure water-blasting or chemical dissolution. The casting are then cut and separated from the runner system. Other post-casting operations (e.g. heat treatment, surface treatment or coating, hipping) can be carried out, according to customer demands.

Here, we focus on the first step (preparation of the die for wax injection and extraction of the pattern from the die) which is a labour demanding operation that has a big impact on the final cost of the product. Specifically, the operation consists of the following steps: (i) mount the die; (ii) inject the wax; (iii) open the die and remove the wax; (iv) repeat the cycle for a new pattern starting back from step (i). The most critical sub-operation is the opening of the die because it has a big impact on the quality of the pattern. In this context, the involvement of a collaborative robot has been envisaged to help the operator in the *assembly/disassembly* operation.

Due to the small size of the dies and the type of operations done by the worker to remove the metallic parts of the die, it is very complex for the robot and the worker to operate on the die simultaneously. However, both of them can cooperate in the assembly/disassembly operation. Once the injection process has finished, the die is taken to the workbench by the worker. The robot and the worker unscrew the bolts of the top cover. There are nine bolts, the robot starts removing those closer to it, and the worker the rest. The robot unscrews the bolts on the cover by means of a pneumatic screwdriver. The worker removes the top cover and leaves it on the assembly area (a virtual

[5] `http://www.fourbythree.eu`

zone that will be used for the re-assembly of the die). The worker turns the die to remove the bottom die cover. The robot unscrews the bolts on the bottom cover by means of a pneumatic screwdriver. Meanwhile the operator unscrews and remove the threaded pins from the two lateral sides to release the inserts. The worker starts removing the metallic inserts from the die and leaves them on the table. Meanwhile, the robot tightens the parts to be assembled/reassembled together screwing bolts. The worker re-builds the die. The worker and the robot screw the closing covers. Thus the human and the robot must collaborate to perform assembly/disassembly on the same die by suitably handling different parts of the die and screwing/unscrewing bolts. Specifically, the human worker has the role of leader of the process while the robot has the role of subordinate with some autonomy. Moreover, the robot must be able to manage a screwdriver device and monitor the human location and its activities.

4.2 An Engineering & Control Architecture for HRC

Given the HRC scenarios described above, many features and constraints must be considered by the envisaged control architecture in order to realize an effective, robust and safe collaboration. Such architecture must implement a well suited tradeoff among the requirements of the different stakeholders involved into the production process, i.e., a *Production Engineer*, a *Knowledge Engineer* and a *Human Worker* in addition to the specific *Robot* requirements. The *Production Engineer* is the expert of the production needs and specifies operational requirements of the different processes that can be performed. The *Knowledge Engineer* knows the features of the robot and of the specific working environment and, therefore, is responsible to model the production processes according to specified operational requirements. The *Human Worker* and the *Robot* are the main actors that actually carry out the production tasks to achieve the production process. In general, several production processes can be performed within a factory. Each process consists of a set of *tasks* that must be executed according to some operational requirements. The perspective pursued here is the following: a *Worker* and a *Robot* represent two *autonomous agents* capable of executing different types of task. Some tasks can be executed only by the human, some tasks can be executed only by the robot and some tasks can be executed by both the human and the robot. Thus, given a particular process, the control system is responsible for synthesizing the set of needed tasks to complete the working process, assigning tasks to the human and to the robot and guaranteeing to *robustly and safely* executing them.

Figure 3 shows the envisaged FOURBYTHREE Engineering & Control Architecture [32] developed for flexible human-robot collaboration and implemented by means of PLATINUM and KEEN. The architecture shows the elements and the actors involved within the control loop as well as their relationships. Specifically, the labeled arrows describe all the phases of the *control process* starting from domain modeling up to physical task execution. The *FbT Engineering Environment* relies on KEEN (*Knowledge Engineering ENvironment*) [16] to support domain experts in the design of the control model exploited by the *FbT Controller* to coordinate the human and the robot tasks. Specifically, the *FbT Engineering Environment* allows the *Production Engineer* and/or the *FbT Knowledge Engineer* to model the working environment and the production processes without knowing in details the specific planning and execution technology

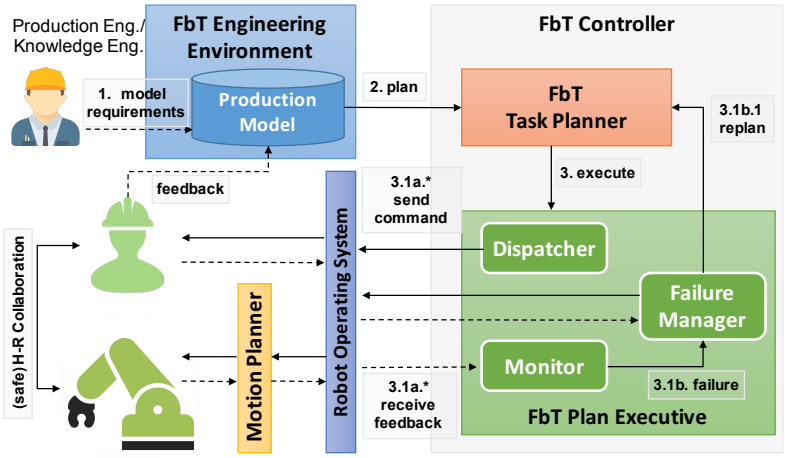

Fig. 3: The FOURBYTHREE Engineering & Control architecture

utilized. Once the model is defined, the *FbT Task Planner* synthesizes a temporal flexible plan assigning tasks to the human and to the robot and the *FbT Plan Executive* executes such plans in order to achieve the production goals. Both the *FbT Task Planner* and the *FbT Plan Executive* rely on PLATINUm. Specifically, the developed task planner is capable of generating temporally robust plan by dealing with *temporal uncertainty* at solving time. This is crucial in the considered scenarios where a human must tightly cooperate with a robot. Indeed, a human is *uncontrollable* and his/her behavior may affect also the behavior of the robot from the control perspective. Thus, the *Human* is modeled as an autonomous and completely *uncontrollable* agent whose behavior may affect the behavior of the *Robot* which is modeled as a *partially controllable* agent.

A plan is executed by *dispatching* commands to the robot and to the human and by receiving *feedbacks* through dedicated communication channels implemented on ROS[6]. The *FbT Plan Executive* realizes a closed-loop control process which puts the human-in-the-loop. Broadly speaking, the executive is capable of dynamically adapting a task plan (i.e., robot task execution) according to the detected behavior of the human. Thus, the executive can temporally adapt a task plan by *absorbing* execution delays and generate a new plan through *replanning* only if strictly needed. Replanning allows the executive to manage exogenous events the plan cannot capture like e.g., a failure of a robot actuator or a human task whose duration is longer than expected and synthesize a new (adapted) plan which tries to complete the execution of the process. It is worth pointing out that the integration of *temporal uncertainty* at both planning and execution time makes the control process more *robust* than classical approaches in the literature e.g., T-REX [8] or IXTET-EXEC [9], limiting the need for replanning.

[6] http://www.ros.org/

4.3 The FOURBYTHREE Controller

The *FbT Controller* is the element responsible to actually carry out production processes and to coordinate the robot and the human. The synthesized tasks and the coordination of the human and the robot must follow the operational requirements specified by the *Production Engineer* and encoded into the *domain model* through KEEN. As Figure 3 shows, the controller is composed by the *FbT Task Planner* and the *FbT Plan Executive* both relying on the timeline-based formalisms. The *FbT Task Planner* is responsible for generating the set of tasks needed to perform the production processes according to the desired requirements. In HRC scenarios, it is necessary to guarantee the safety of the human without penalizing the productivity of the factory. The task planner is in charge of finding a tradeoff between performance and safety and therefore there are several features to take into account when synthesizing plans.

The planning model can be characterized according to three different levels of abstraction: (i) the *supervision level*; (ii) the *coordination level*; (iii) the *implementation level*. In the *supervision level*, the task planner has to decide the set of tasks needed to execute the production process by modeling the operational requirements specified by the *Production Engineer*. In the *coordination level*, the task planner has to decide who, between the human and the robot, must perform each task harmonizing the activities of both. In this context, the human and the robot are modeled as *two autonomous agents* capable of executing some types of task. Given a production process, some tasks can be performed only by the human, some tasks can be performed only by the robot and some tasks are *free* to be performed either by the human or by the robot. This choice-point represents the main *branching factor* of the task planning process. It can affect the *quality* of the collaboration and the efficiency of processes. Finally, in the *implementation level*, the task planner has to decide the operations the robot must perform in order to execute the assigned tasks. According to the particular type of collaboration decided at coordination level, the task planner decides the most appropriate *execution modality* of the tasks of the robot in order to preserve the safety of the human.

Figure 4 (automatically generated by KEEN) shows an example of a timeline-based planning model for the collaborative assembly scenario. The model is hierarchically organized according to the three levels of abstraction identified (i.e., *supervision*, *coordination*, *implementation*). The *ALFA* (i.e., the name of the pilot plant) and *AssemblyProcess* state variables compose the *supervision level* of the model. These variables characterize the considered production context in terms of tasks that can be executed. The *AssemblyProcess* specifies the set of *high-level tasks* needed to complete the process and the related operational requirements. For example, the *RemoveTopCover* and *RemoveBottomCover* values in *AssemblyProcess* represent high-level tasks modeling part of the assembly/disassembly procedure. Notice that no task assignment is performed at this level of abstraction. The *Human*, *RobotController* and *CollaborationType* state variables compose the *coordination level* of the model. Specifically, the *Human* and *RobotController* state variables model the *low-level* tasks the human and the robot agents can perform over time. For example, the *Screw* or *Unscrew* values of *Human* and *RobotController* state variables model the capability of both *agents* of performing screwing operations. Instead, *RemovePart* or *Rotate* values of the *Human* state variables model *critical operations* that only the human is allowed to perform. The *CollaborationType*

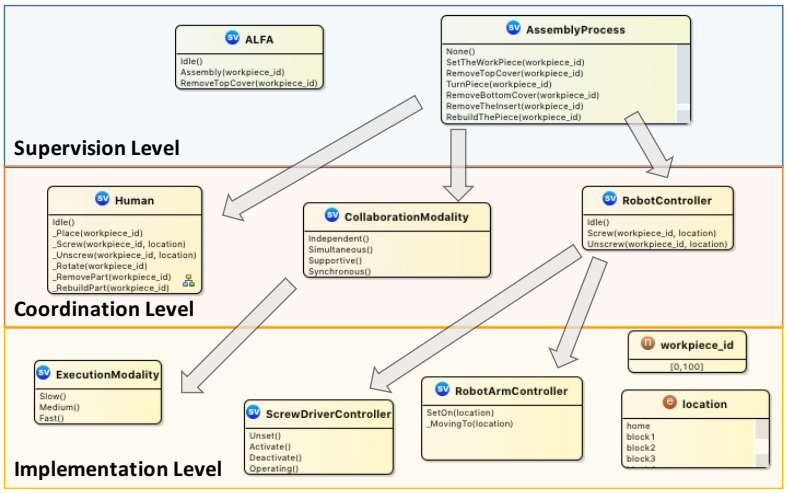

Fig. 4: A graphical overview of the hierarchical P&S model for collaborative assembly generated by KEEN.

state variable models the possible types of human-robot collaboration within the execution of the tasks of the desired process. The supervision and coordination layers are connected by a set of synchronization rules that specify decomposition constraints of *high-level tasks* in terms of *low-level tasks*. These rules specify how the tasks composing the process can be performed in collaboration by the human and the robot. Namely, these rules describe the possible task assignments between the human and the robot and specify the collaboration modalities suited for human-robot interactions.

The *RobotArmController*, *ScrewDriverController* and *ExecutionModality* state variables constitute the *implementation level* of the model. These variables represent the physical and/or logical elements composing the production environment the system must directly interact with. The *RobotArmController* together with the *ExecutionModality* model the robotic arm. They represent the functional control interface of the robot provided by the integrated *motion planner* (see Figure 3). Specifically, the *RobotArmController* models the motion tasks the robot can perform while, the *ExecutionModality* models the type of trajectory that must be used to perform the motion. The coordination and implementation layers are connected by another set of synchronization rules that specify how the robot must execute the assigned tasks. A particular execution modality of robot motions is selected according to the expected collaboration modality in the coordination layer.

4.4 Implementation with a Real Robot

The Engineering & Control Architecture described above was deployed in a manufacturing case study integrating the task planning technology described above with a motion planning system for industrial robots [33]. The reference selected application is a human-robot collaborative environment for the preparation of the load/unload station (LUS) of a flexible manufacturing system (FMS). At the LUS, machined parts and raw

parts have to be unmounted and mounted on ad-hoc fixturing systems, called pallet, by a worker and a robot in order to be machined by the FMS. With the aim to increase productivity and grant human ergonomics and safety, robot trajectories and task allocation have to be respectively adequately designed and planned.

Thus, PLATINUM and KEEN features were 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 [34], an empirical evaluation is provided in order to assess the overall productivity of the HRC cell while increasing the involvement of the robots. The idea is to gradually make free a set of tasks originally preallocated to the human, so to increase the number of degrees of freedom of the task planner during the minimization of the assembly time. The results show the effectiveness of the control architecture in finding well suited distribution of tasks between the human and the robot in different scenarios with an increasing workload for the control system. Indeed, the total assembly time was reduced of 65% (from 259s to 169s) and the percentage of tasks assigned by the controller to the robot moved from 25% to 65%. Thus, PLATINUM and KEEN resulted capable of increasing the productivity of the production process without affecting the safety of the operator. It is worth underscoring that the outcome of this integration constitutes the technological basis on which a new research project development is undergoing, i.e., the ShareWork project[7] funded by the European Commission within the Factories of the Future area.

5 Conclusions

A research initiative has been started to investigate the possible integration of a timeline-based planning framework (APSI-TRF [4]) and V&V techniques based on Timed Game Automata (TGA) [5] to automatically synthesize a robot controller that guarantees robustness and safety properties [6, 7]. This chapter provided an overview of the results collected in the last decade concerning the development of a task planning and execution framework (PLATINUM) [14, 15] and its integration with a Knowledge Engineering ENvironment (KEEN) describing its deployment for developing safe and effective solutions for manufacturing HRC scenarios. KEEN [16] is a knowledge engineering environment with Verification & Validation features based on Timed Game Automata model checking. PLATINUM is a timeline-based planning systems capable of supporting flexible temporal planning and execution with uncertainty. A brief overview on recent results about the development of a task planning and execution technology

[7] `https://sharework-project.eu/`

deployed in realistic manufacturing scenarios was presented. The research agenda is far from being completed. The long-term goal is to realize a robust task planning system enabling flexible, safe and efficient HRC with a more tight integration of P&S technology with Knowledge Engineering solutions and V&V techniques is a key element to synthesize more effective, robust and reliable safety critical systems in robotics [2].

Acnkowledgements

Amedeo Cesta, Andrea Orlandini and Alessandro Umbrico wish to acknowledge the support by the European Commission and the ShareWork project (H2020 – Factories of the Future – G.A. nr. 820807).

References

1. 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) (2016) 433 – 436
2. Bensalem, S., Havelund, K., Orlandini, A.: Verification and validation meet planning and scheduling. International Journal on Software Tools for Technology Transfer **16**(1) (2014) 1–12
3. La Viola, C., Cesta, A., Orlandini, A., Umbrico, A.: Ros-tiplex: How to make experts in a.i. planning and robotics talk together and be happy. In: RO-MAN 2018 - 27th IEEE International Symposium on Robot and Human Interactive Communication, Institute of Electrical and Electronics Engineers Inc. (To Appear) (2019)
4. Cesta, A., Cortellessa, G., Fratini, S., Oddi, A.: 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)
5. Maler, O., Pnueli, A., Sifakis, J.: On the Synthesis of Discrete Controllers for Timed Systems. In: STACS. LNCS, Springer (1995) 229–242
6. 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) (2010) 299–318
7. Orlandini, A., Suriano, M., Cesta, A., Finzi, A.: Controller synthesis for safety critical planning. In: IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI 2013), IEEE (2013) 306–313
8. Py, F., Rajan, K., McGann, C.: A systematic agent framework for situated autonomous systems. In: AAMAS. (2010) 583–590
9. Lemai, S., Ingrand, F.: Interleaving Temporal Planning and Execution in Robotics Domains. In: AAAI-04. (2004) 617–622
10. 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)
11. Ghallab, M., Laruelle, H.: Representation and control in ixtet, a temporal planner. In: 2nd Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS). (1994) 61–67
12. Morris, P.H., Muscettola, N.: Temporal Dynamic Controllability Revisited. In: Proc. of AAAI 2005. (2005) 1193–1198

13. 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)
14. Umbrico, A., Cesta, A., Cialdea Mayer, M., Orlandini, A.: PLATINUM: A new Framework for Planning and Acting. In: AI*IA 2016 Advances in Artificial Intelligence: XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, November 29 – December 1, 2016, Proceedings, Springer International Publishing (2017) 508–522
15. Umbrico, A., Cesta, A., Cialdea Mayer, M., Orlandini, A.: Integrating resource management and timeline-based planning. In: The 28th International Conference on Automated Planning and Scheduling (ICAPS). (2018)
16. Orlandini, A., Bernardi, G., Cesta, A., Finzi, A.: Planning meets verification and validation in a knowledge engineering environment. Intelligenza Artificiale **8**(1) (2014) 87–100
17. Cesta, A., Cortellessa, G., Fratini, S., Oddi, A., Policella, N.: An Innovative Product for Space Mission Planning: An A Posteriori Evaluation. In: ICAPS. (2007) 57–64
18. Jonsson, A., Morris, P., Muscettola, N., Rajan, K., Smith, B.: Planning in Interplanetary Space: Theory and Practice. In: AIPS-00. Proceedings of the Fifth Int. Conf. on AI Planning and Scheduling. (2000)
19. Muscettola, N.: HSTS: Integrating Planning and Scheduling. In Zweben, M. and Fox, M.S., ed.: Intelligent Scheduling. Morgan Kauffmann (1994)
20. Vidal, T., Fargier, H.: Handling Contingency in Temporal Constraint Networks: From Consistency To Controllabilities. JETAI **11**(1) (1999) 23–45
21. Cialdea Mayer, M., Orlandini, A., Umbrico, A.: Planning and execution with flexible timelines: a formal account. Acta Informatica **53**(6-8) (2016) 649–680
22. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11) (1983) 832–843
23. Umbrico, A., Cesta, A., Cortellessa, G., Orlandini, A.: A goal triggering mechanism for continuous human-robot interaction. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **11298 LNAI** (2018) 460–473
24. Cesta, A., Finzi, A., Fratini, S., Orlandini, A., Tronci, E.: Flexible Timeline-Based Plan Verification. In: KI 2009: Advances in Artificial Intelligence. Volume 5803 of LNAI. (2009)
25. Cesta, A., Finzi, A., Fratini, S., Orlandini, A., Tronci, E.: Analyzing Flexible Timeline Plan. In: ECAI 2010. Proceedings of the 19th European Conference on Artificial Intelligence. Volume 215., IOS Press (2010)
26. Orlandini, A., Finzi, A., Cesta, A., Fratini, S.: Tga-based controllers for flexible plan execution. In: KI 2011: Advances in Artificial Intelligence, 34th Annual German Conference on AI. Volume 7006 of Lecture Notes in Computer Science., Springer (2011) 233–245
27. Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K., Lime, D.: UPPAAL-TIGA: Time for playing games! In: Proc. of CAV-07. Number 4590 in LNCS, Springer (2007) 121–125
28. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a Nutshell. International Journal on Software Tools for Technology Transfer **1**(1-2) (1997) 134–152
29. Marvel, J.A., Falco, J., Marstio, I.: Characterizing task-based human-robot collaboration safety in manufacturing. IEEE Trans. Systems, Man, and Cybernetics: Systems **45**(2) (2015) 260–275
30. Helms, E., Schraft, R.D., Hagele, M.: rob@work: Robot assistant in industrial environments. In: Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication. (2002) 399–404

31. 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). (Sept 2016) 1–8
32. Cesta, A., Orlandini, A., Umbrico, A.: Fostering robust human-robot collaboration through ai task planning. Procedia CIRP **72** (2018) 1045 – 1050 51st CIRP Conference on Manufacturing Systems.
33. 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) (2016) 57 – 60
34. Pellegrinelli, S., Orlandini, A., Pedrocchi, N., Umbrico, A., Tolio, T.: Motion planning and scheduling for human and industrial-robot collaboration. CIRP Annals - Manufacturing Technology **66** (2017) 1–4