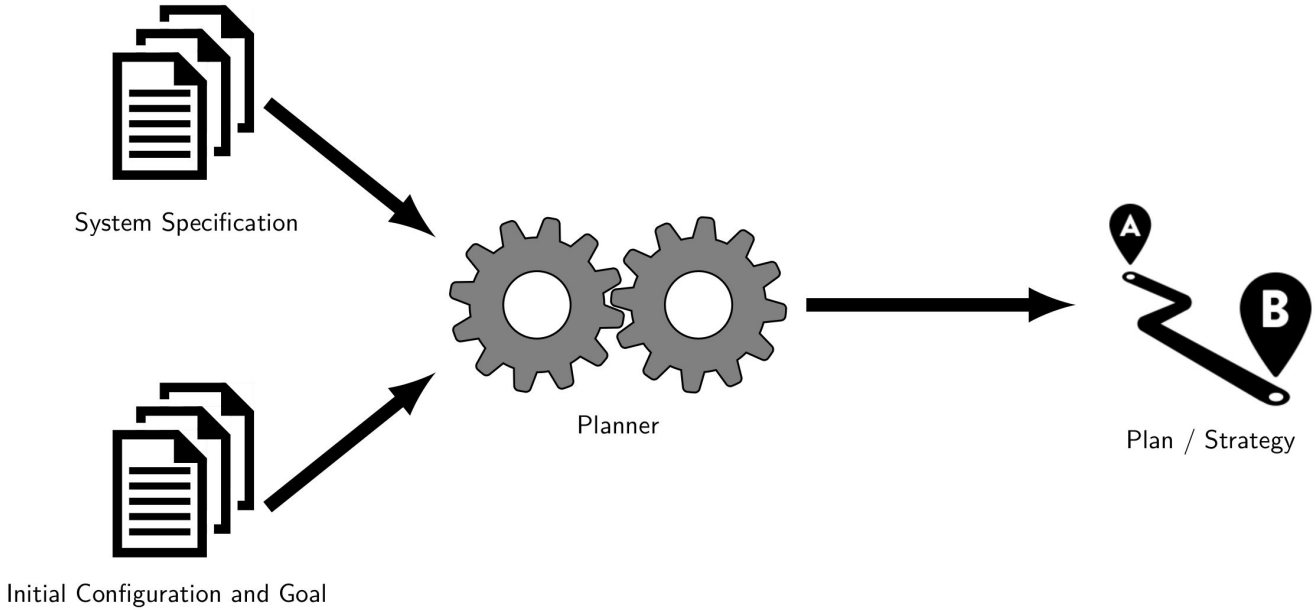
AIPlan4EU-AI4Europe Integration: User Journeys and Implementation Ideas

aiplan4eu-project.eu

The AIPlan4EU project is funded by the European Commission - H2020 research and innovation programme under grant agreement No 101016442

Automated Planning in a Nutshell

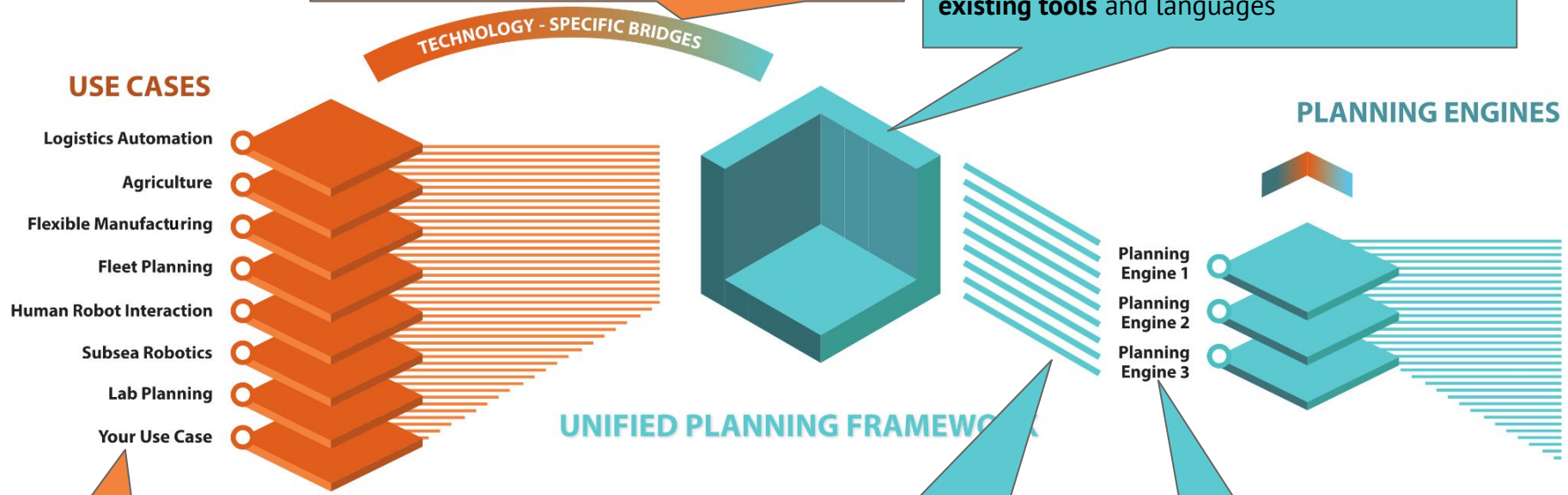
Given a **model of a system** and a **goal to be reached**, find a **course of actions** to drive the system to the goal



Vision

The API offered by the UPF is integrated by a **collection of reusable components** that are specific to a certain technology (not use-case specific). E.g. Integration in ROS, Integration in a WMS, ...

The UPF is a **reusable, planner-agnostic** Python library, offering an **abstraction layer** for diverse planning engines and **interoperability with existing tools** and languages



Cascade-funding to elicit **additional use-cases** and develop **demonstrators**

The UPF has a notion of "**Operation Mode**" (OM):

- OM examples include "**OneshotPlanning**", "**PlanValidation**" and "**PlanRepair**"
- Each engine declares which operation modes it supports

"Planning Engines" is a general word: plan **generators**, plan **validators**, **visualizations**...

UP Library General Look-and-Feel

```
from unified_planning.shortcuts import *

x = Fluent('x', BoolType()) # Finite- and infinite-domain state variables are supported

a = InstantaneousAction('a')
a.add_precondition(Not(x))
a.add_effect(x, True)

# Problem is defined in code, possibly using data from other ICT systems
problem = Problem('basic')
problem.add_fluent(x)
problem.add_action(a)
problem.set_initial_value(x, False)
problem.add_goal(x)

# Problem features are automatically detected and planner is automatically selected
# among the installed engines
with OneshotPlanner(problem_kind=problem.kind()) as planner:
    # The same API works for any planner supporting the Oneshot Operation Mode
    result = planner.solve(problem)

    # Results are inspectable data structures, easy to use for interoperability
    print('\n'.join(str(x) for x in result.plan.actions()))
```

Live demo: https://bit.ly/3rZlUHE

AI Assets and Case Studies

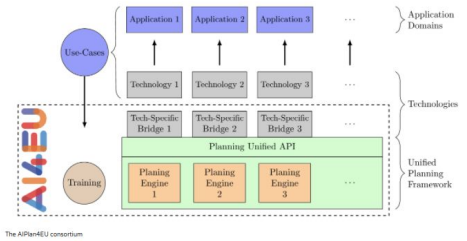
AI4EU | AI Community | Business & Industry | Research | Education | Ethics | Services | News & Events

Home > Research > AI Assets > Unified Planning Framework

Unified Planning Framework

The Unified Planning Framework (UPF) library makes it easy to formulate planning problems and to invoke automated planners.

Library | GitHub page



The AIPlan4EU consortium

Contact Details
Andrea Micheli, Fondazione
Bruno Kessler (amicheli@fbk.eu)

License
Apache License 2.0 (Apache-2.0)

Main Characteristic

Define problems in a simple, intuitive, and planner independent way

- Solve your planning problems using one of the native solvers, or by using any PDDL planner
- Dump your problems in PDDL (or ANML) format
- Parse PDDL problem formulations
- Simplification, grounding, removal of conditional effects and many other transformations are available
- and more...

The UPF library is being developed by the

Detailed Description

The purpose of the library is to provide an abstraction layer for planning technology allowing a user to specify planning problems in a planner independent way and then use one of the available planning engines installed on the system. The library is implemented as a Python package offering high level API to specify planning problems and to invoke planning engineers. Moreover, the library offers functionalities for transforming and simplifying planning problems and to parse problems from existing formal languages.
The library is being developed publicly under a

AI4EU | AI Community | Business & Industry | Research | Education | Ethics | Services | News & Events

Home > > Planning for Logistics Automation

Planning for Logistics Automation

Magazino is an innovative company producing robots for warehouse intra-logistics

Developed by
Magazino GmbH

Business Category
Manufacturing | Transportation

Technical Category
Planning and scheduling
Robotics and automation

tenorth@magazino.eu
gemignani@magazino.eu

https://www.aiplan4eu-project.eu/

In Magazino, each robot is given a prioritized list of jobs to perform, and each of them is associated with a hand-written plan, encoded using a behavior tree formalism.

This solution has a few drawbacks: handwriting of the behavior trees requires expert knowledge and quite some brain power to be carried out without mistakes. This makes the writing tedious and the maintenance and update complex and error prone. For example, in addition to take into account the possible failures and unexpected events, such plans need to be crafted considering that the executor might get restarted at any point in time, losing and thus needing to reconstruct the world and internal states. Finally, the behavior trees are typically rather complex and they are not suitable to be analyzed or manipulated by some high-level reasoning mechanism.

In this context, planning techniques can be used to guide the user to design the complex plans required to accomplish the tasks the robot is given, and to follow their execution taking into account the automatic recovery from unforeseen circumstances, e.g., after a failure has occurred.

Part1: What do we want to achieve

(User Journeys)

Personas

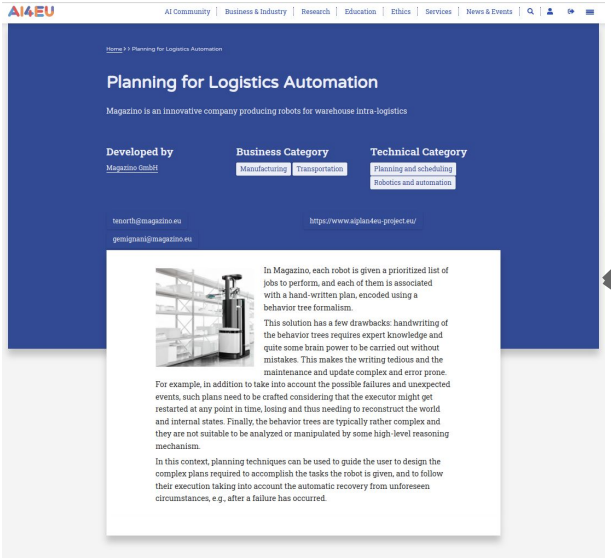
In the platform, there are different kinds of users, at the very minimum we distinguish:

- **Business user**
  - Not interested in the technicalities of the solutions
    - Cannot understand code
  - Focuses on the impact and the networking
    - Interested in use-cases and evaluations
- **Technical user** (either researcher or CTO or tech-savvy):
  - Wants to see how solutions work
    - Can understand code
  - Focuses on technical feasibility and performance
    - Considers hardware and software requirements
  - Interested in UPF and TSBs

Disclaimers:

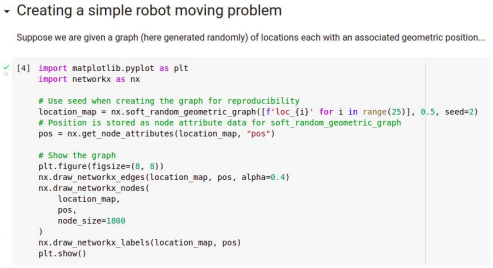
- This is a very rough characterization (a more principled user study would be needed at a platform level), but it serves to draft the design that will follow.
- At the moment, content is not differentiated according to the user profile (this choice is more transversal), so we plan to offer both technical and business content, but clearly differentiate it visually

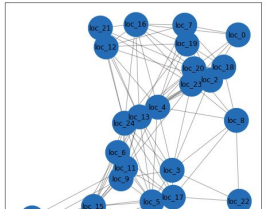
User Experience: General Idea (1)



AIPlan4EU Use-case page on AI4EU CMS

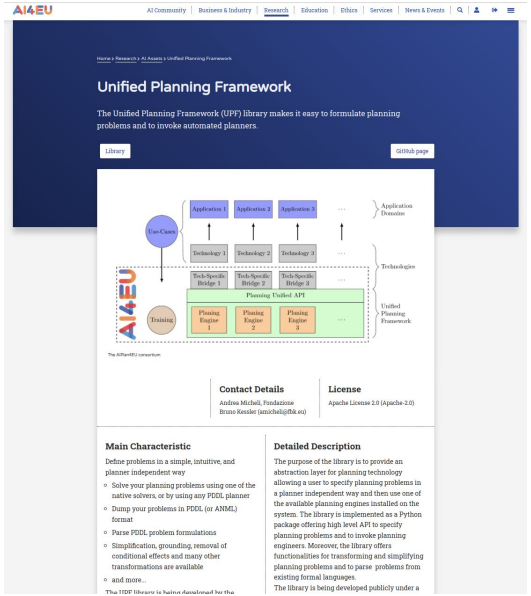
KPIs and evaluation results for the business user





Live, interactive demo for the technical user

User Experience: General Idea (2)



AIPlan4EU-related asset page on AI4EU CMS

Links to use-cases, evaluation KPIs and Companies using it for the business user

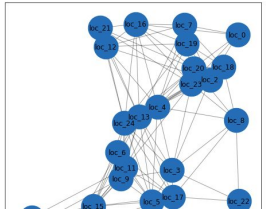
Creating a simple robot moving problem

Suppose we are given a graph (here generated randomly) of locations each with an associated geometric position...

```
[4] import matplotlib.pyplot as plt
    import networkx as nx

    # Use seed when creating the graph for reproducibility
    location_map = nx.soft_random_geometric_graph([f'loc_{i}' for i in range(25)], 0.5, seed=2)
    # Position is stored as node attribute data for soft_random_geometric_graph
    pos = nx.get_node_attributes(location_map, "pos")

    # Show the graph
    plt.figure(figsize=(8, 8))
    nx.draw_networkx_edges(location_map, pos, alpha=0.4)
    nx.draw_networkx_nodes(
        location_map,
        pos,
        node_size=1800
    )
    nx.draw_networkx_labels(location_map, pos)
    plt.show()
```

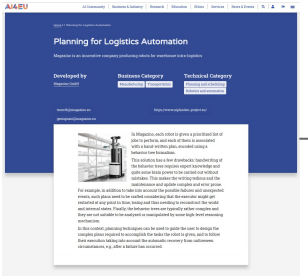


Live, interactive demos for the technical user

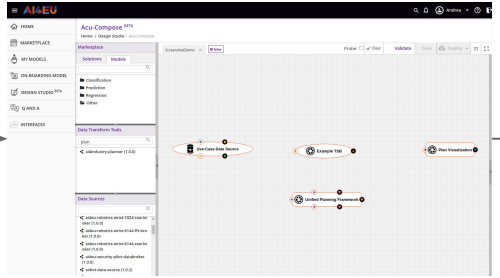
Part2: Implementation on top of AI4Europe

Integration in AI4Europe

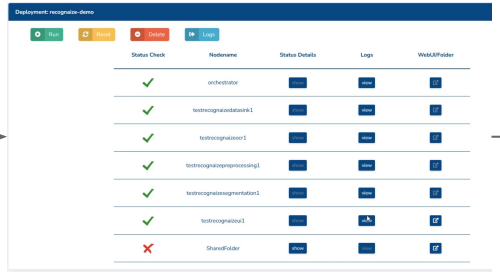
- The business journey is easy: just some (appropriately structured and tagged) CMS pages are needed
- The technical journey requires to execute the demo code (e.g. a python notebook). Three AI4EU parts involved:
  - CMS
    - To present “static” content (assets, use-cases, KPIs, evaluations)
  - AI4EU Experiments
    - To create the demo as pipelines for execution
  - AI4EU Playground
    - To execute the pipelines and present the interactive GUIs (e.g. Python Notebooks)



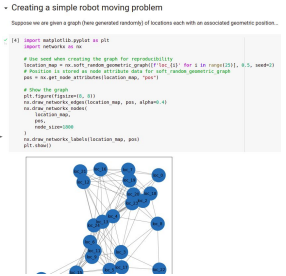
Use-case CMS page



AI4EU Experiments pipeline



AI4EU Playground



AIPlan4EU Python notebook

Next Actions (and What We Can Provide)

- We are currently working on an example pipeline using our planning components (by 06/22)
  - A simple web GUI that allows the creation of a maze a problem communicating (via GRPC) to a planner and a simulator
  - Uses AI4EU Experiments pipelines to showcase UPF + Technology-Specific Bridge
- We will create a GUI component serving a pre-configured Python Notebook with our library (by 09/22)
  - Zero-installation, try-before-you-buy
  - Remote execution on playground
- We will add support for recursive data-structures and enumerative types in AI4EU Experiments (by 01/23)
  - We will “hire” Peter Schüller for this
- Reusability: after creating our pilot pipelines, we could create templates for everyone to use

Some Usability Considerations

- AI4EU Experiments is a rather articulated service; it would be better to directly bring the user from CMS to Playground (or even better to the Python Notebook / GUI)
  - We can offer the possibility to inspect/edit Pipelines for advanced users
  - A “Waiting Page” (similar to Binder) would be useful to keep the user waiting while the underlying system does its job
- How to maintain consistency in the offered GUIs?
  - Create a custom theme / add logos…
- Any feedback and support on these points is greatly appreciated :)

AIPLAN 4EU

Thanks for your attention!

aiplan4eu-project.eu