Planificador inteligente preliminar basado en PDDL para ensamble de piezas a nivel industrial.

PDDL based preliminary intelligent planner for industrial-level parts assembly.

Arturo Fernando Cruz Reyes (1).
Estudiante Tecnológico Nacional de México/I. T. de Tuxtla Gutiérrez.
L20270798@tuxtla.tecnm.mx.

Guillen Meza Mauricio Antonio (2). Estudiante Tecnológico Nacional de México/I. T. de Tuxtla Gutiérrez,
L20270281@tuxtla.tecnm.mx.

Héctor Guerra Crespo* (3). Tecnológico Nacional de México/I. T. de Tuxtla Gutiérrez. hector.gc@tuxtla.tecnm.mx.

Luis Tomás García Andrade (4). Tecnológico Nacional de México/I. T. de Tuxtla Gutiérrez,
luis.ga1@tuxtla.tecnm.mx.

Néstor Antonio Morales Navarro (5), Tecnológico Nacional de México/I. T. de Tuxtla Gutiérrez,
nestor.mn@tuxtla.tecnm.mx.

*corresponding author.

Artículo recibido en noviembre 26, 2025; aceptado en diciembre 05, 2025.

Resumen.

Este documento presenta el desarrollo preliminar de un planificador inteligente basado en PDDL para automatizar la secuencia de ensamblaje de piezas en un entorno industrial. La planificación automática, como rama de la inteligencia artificial, permite generar planes que transforman un estado inicial en uno objetivo mediante acciones formalmente definidas. El proyecto toma como caso de estudio la planta Yazaki Tuxtla 2, dedicada a la manufactura de arneses automotrices, donde la complejidad operativa justifica la necesidad de un planificador formal. El método incluye la definición de requisitos, modelado en PDDL, validación conceptual y pruebas en el entorno editor.planning.domains. El dominio modela una línea de producción con cuatro etapas: cortar, taladrar, pulir y empacar, cada una representada mediante predicados, precondiciones y efectos en estilo STRIPS. Se especifica un ejemplo completo con estado inicial, objetivo y el plan óptimo generado. El trabajo demuestra que PDDL es capaz de representar secuencias industriales reales y constituye la base para futuras implementaciones de planificación integral en líneas de producción automotriz.

Palabras claves: Industria, PDDL, planificación.

Abstract.

This document presents the preliminary development of an intelligent planner based on PDDL to automate the assembly sequence of parts in an industrial environment. Automatic planning, as a branch of artificial intelligence, allows for the generation of plans that transform an initial state into a target state through formally defined actions. The project uses the Yazaki Tuxtla 2 plant, dedicated to the manufacture of automotive wiring harnesses, as a case study, where operational complexity justifies the need for a formal planner. The method includes requirements

definition, PDDL modeling, conceptual validation, and testing in the editor.planning.domains environment. The domain models a production line with four stages: cutting, drilling, polishing, and packaging, each represented by predicates, preconditions, and effects in the STRIPS style. A complete example is specified with the initial state, target state, and the generated optimal plan. This work demonstrates that PDDL is capable of representing real industrial sequences and provides a foundation for future implementations of comprehensive planning in automotive production lines.

Keywords: Industry, PDDL, planning.

1. Introducción.

La planificación automática constituye una disciplina central dentro de la inteligencia artificial que se ocupa de generar secuencias de acciones (planes) que permiten a un agente alcanzar objetivos deseados bajo restricciones temporales, de recursos y operativas. Desde sus orígenes, la comunidad de planificación ha trabajado en la formalización de lenguajes y modelos que permitan describir dominios y problemas de forma independiente del algoritmo, facilitando la comparación, validación y transferencia de soluciones entre contextos aplicados.

Dentro de la planificación automática el lenguaje de definición de dominios en planificación (PDDL, Planning Domain Definition Language) surgió como estándar para describir dominios y problemas de planificación, y sus sucesivas versiones (por ejemplo PDDL 2.1) añadieron soporte para fenómenos temporales y recursos numéricos, elementos esenciales para modelar procesos industriales reales (McDermott D. et al, 1998).

En el ámbito industrial, en particular industria automotriz, la planificación automática juega un papel crítico en tareas como secuenciación de operaciones, asignación de recursos, replanteo on-the-fly ante fallos de línea y coordinación de vehículos y manipuladores. La complejidad de las líneas de producción automotriz (multiple estaciones, restricción de herramientas, tiempos de acoplamiento, y requisitos de calidad) exige representaciones que capten duración de acciones, recursos numéricos y condiciones continuas; por ello, extensiones del lenguaje PDDL que permiten durativas y expresiones numéricas han sido empleadas para acercar la planificación académica a problemas industriales. (Fox M., 2003):

El desarrollo de este documento se inspira en el trabajo que realiza la planta Yazaki Tuxtla 2 (figura 2), dedicada a la manufactura de arneses automotrices (figura 1).

Figura 1. Arnés eléctrico para vehículos híbridos y eléctricos de las marcas GM y Subaru. Fuente sitio web empresa japonesa Yazaki. Recuperado diciembre 2024.

Figura 2. Planta Yazaki Tuxtla 2. Fotografia diciembre 2024.

La planta cuenta con 6 líneas de producción.
Línea modelo. Estandariza la fabricación del arnés. Esta lìnea es la piloto en el desarrollo de este trabajo.
Cada línea cuenta con 4 estaciones subensamble, direccionado, encintado y línea final.
Cada línea tienen entre 12 y 20 estaciones de subensambles dependiendo de la complejidad del arnés.

2. Métodos.

En esta etapa del desarrollo el propósito es desarrollar la parte del planificador que contenga al menos la secuencia de tres piezas que ilustre el potencial para implementar de manera real una línea completa de producciòn.

1. Análisis de Requisitos.
   Definir el componente inicial (pieza) del planificador
   Establecer los recursos necesarios: herramientas, componentes, estaciones de ensamblaje.
   Incorporar restricciones de ensamblaje (ej. secuencia lógica, espacio físico, tiempo de operación).
   Considerar restricciones de seguridad.
   Determinar la tecnología de implementación.

2. Modelo PDDL del planificador, pieza inicial del ensamblaje.
   Definir estado inicial
   Definir estado final
   Acciones (parámetros, precondiciones, efectos)

3. Validación preliminar del planificador
   Validar el modelo conceptualmente
   Validar el modelo en un editor PDDL
   Crear escenarios de prueba representativos de ensamblajes reales.
   Ajustar el modelo (acciones) de acuerdo a los escenarios.

Breve historia de la planificación automática.

Planificación concepto general.

La real academia española (RAE, 2025) define planificación como

1. Acción y efecto de planificar.
2. Plan general, metódicamente organizado y frecuentemente de gran amplitud, para obtener un objetivo deter minado, tal como el desarrollo armónico de una ciudad, el desarrollo económico, la investigación científica, el funcionamiento de una industria, etc.

Planificacion automática como rama de la inteligencia artificial se dedica a la generación de secuencias de acciones (planes) que transforman un estado inicial en uno o varios estados objetivo, teniendo en cuenta las restricciones del dominio y los efectos de las acciones. La planificación produce deliberadamente planes antes de su ejecución o durante la ejecución cuando es necesario replantearlos, lo que la hace esencial para sistemas autónomos, robótica, logística y optimización de procesos industriales (Ghallab, 2004).

General Problem Solver (GPS).

Se reconoce como el primer planificador y uno de los primeros programas del área de Inteligencia Artificial desarrollado por Newell y Simon en 1959 (Newell y Simon, 1959). El programa se llamó General Problem Solver (GPS) basado en la técnica means–ends analysis es un método heurístico que consiste en:

- Comparar el estado actual con el estado meta.
- Identificar las diferencias entre ambos.
- Seleccionar una operación (medio) que reduzca alguna diferencia (fin).
- Aplicar la operación para acercarse al objetivo.
- Repetir el proceso hasta alcanzar la meta.
- La técnica busca minimizar la diferencia entre el estado actual y el estado deseado paso a paso.

STRIPS.

Un salto importante en las técnicas de planificación fueron los STRIPS (Stanford Research Institute Problem Solver). Es uno de los modelos más influyentes en la historia de la planificación automática en inteligencia artificial. Fue desarrollado por Richard Fikes y Nils Nilsson en el SRI (Stanford Research Institute) a inicios de los años 1970 (Fikes y Nilsson, 1971).

STRIPS es un formalismo para describir problemas de planificación, es decir, para especificar:

- Estado inicial.
- Estado meta (objetivos).
- Acciones disponibles, cada una con:
- Precondiciones (lo que debe cumplirse para ejecutarla).
- Add list (efectos positivos: lo que añade al estado).
- Delete list (efectos negativos: lo que elimina del estado).

STRIPS se convirtió en el estándar para que los algoritmos de planificación puedan razonar y buscar planes.

STRIPS es importante porque introdujo ideas clave que definieron la planificación moderna como:

- Representación lógica de estados. Los estados se representan con predicados lógicos, lo que permite razonar formalmente.
- Efectos separados en add y delete lists lo que facilita la simulación de acciones y cálculo de estados sucesores.
- Precondiciones explícitas. Permite saber qué acciones son aplicables en un estado.
- Base del lenguaje moderno PDDL. El PDDL (Planning Domain Definition Language) usado hoy en competencias internacionales ICLP está basado directamente en STRIPS.
- Fundó la “planificación clásica”.

Graphplan.

A mediados de los 90 la planificación dió un salto importante con la introducción de Graphplan (Blum A. y Furst M., 1995), GraphPlan es un algoritmo de planificación automática que construye una estructura llamada grafo de planificación (planning graph) para encontrar planes eficientes de manera rápida.

Es importante por que.

- Es más rápido que los planificadores anteriores
- Introduce el uso de mutuas exclusiones (mutex). Sirven para evitar combinaciones imposibles.
- Combina expansión estructurada y búsqueda eficiente.
- Genera planes mínimos en número de pasos.
- Se convirtió en la base de muchos planificadores modernos.
- PDDL adoptó su filosofía y muchas aplicaciones de planificación usan variantes de GraphPlan.

PDDL, lenguaje de definición de dominios para planificación.

La planificación en Inteligencia Artificial por lo general utiliza un lenguaje de modelación llamado PDDL (Planning Domain Definition Language), en la actualidad prácticamente es el lenguaje estándar para la codificación de los dominios de planificación dado a conocer por Drew McDermott en la International Planning Competition (IPC) de 1998 (Mcdermott, 1998).

PDDL está basado en STRIPS y cuenta con las versiones 2.1 (2002), 2.2 (2004), 3.0 (2006) hasta la actual 3.1 y última versión de 2011.

International Planning Competition (IPC).

La IPC (ICAPS, 2025) es una competencia académica internacional, organizada por la comunidad de planificación automática, que reúne a investigadores de todo el mundo para evaluar planificadores (planners) usando un conjunto estandarizado de dominios y problemas expresados en PDDL.

Se realiza aproximadamente cada dos años y está vinculada con el congreso ICAPS (International Conference on Automated Planning and Scheduling).

Objetivos principales de la IPC.

- Establecer un marco de evaluación común para comparar planificadores.
- Proveer métricas estandarizadas que permitan medir rendimiento.
- Evaluar eficiencia, escalabilidad, calidad del plan y tiempo de cómputo.
- Motivar el desarrollo de nuevos algoritmos y enfoques de planificación.
- Impulsar la evolución del lenguaje PDDL, que nació junto con la IPC.

3. Desarrollo.

La implementación del planificador preliminar es a través de la herramienta *editor.planning.domains* del proyecto Education.Planning.Domains (Education.Planning.Domains, 2025).

Etapas en la especificación del planificador.

1. Descripción del dominio.

El dominio considera una línea de producción donde cada pieza debe pasar por una secuencia de etapas o procesos

- Cortado.
- Taladrado.
- Pulido.
- Empacado.

Cada operación solo puede realizarse si la pieza está en el área correspondiente y no ha completado aún ese proceso.

Áreas/departamentos
- Cortar.
- Taladrar.
- Pulir.
- Empacar.

2. Predicados (Predicates).

Predicado	Significado
pieza(p)	p es una pieza industrial.
etapa(p,e)	La pieza p se encuentra lista para el proceso/etapa e.
cortado(p)	La pieza p está cortada.
taladrado(p)	La pieza p está taladrada.
pulido(p)	La pieza p está pulida.
empacado(p)	La pieza p está empacada.
libre(m)	La máquina está disponible.
area(p,a)	La pieza p está en un área específica.

p=pieza
e=etapa
m=máquina
a=área

3. Operadores STRIPS (Acciones).

A continuación se listan acciones STRIPS para cada proceso. Cada acción incluye: precondiciones, efectos (add y delete).

Acción 1: cortar_pieza
Action: cortar-pieza(?p)
Preconditions:
  pieza(?p)
  etapa(?p, cortado)
  area(?p, área-cortar)
  libre(máquina-cortar)
Add Effects:
  cortado(?p)
  etapa(?p, taladrado)
Delete Effects:
  etapa(?p, cortado)

Cortar pieza es el proceso inicial y sus precondiciones es que la pieza se encuente en el área de cortado y la máquina de cortar esté libre. Su efecto es que la pieza pase a a la etapa de taladrado y la pieza se defina como cortada.

Acción 2: taladrar_pieza
Action: taladrar-pieza(?p)
Preconditions:
pieza(?p)
cortado(?p)
etapa(?p, taladrado)
area(?p, área-taladrar)
libre(máquina-taladrar)
Add Effects:
taladrado(?p)
etapa(?p, pulido)
Delete Effects:
etapa(?p, taladrado)

Taladrar pieza es el segundo proceso y sus precondiciones es que la pieza se encuente en el área de taladrado, que este cortada y la máquina de taladrar esté libre. Su efecto es que la pieza pase a la etapa de pulido y la pieza se defina como taladrada.

Acción 3: pulir_pieza
Action: pulir-pieza(?p)
Preconditions:
pieza(?p)
taladrado(?p)
etapa(?p, pulido)
area(?p, área-pulir)
libre(maquina-pulir)
Add Effects:
pulido(?p)
etapa(?p, empacado)
Delete Effects:
etapa(?p, pulido)

Pulir pieza es el tercer proceso y sus precondiciones es que la pieza se encuente en el área de pulido, que este taladrada y la máquina de pulir esté libre. Su efecto es que la pieza pase a la etapa de empacado y la pieza se defina como pulida.

Acción 4: empacar_pieza
Action: empacar-pieza(?p)
Preconditions:
pieza(?p)
pulido(?p)
etapa(?p, empacar)
area(?p, área-empacar)
libre(máquina-empacar)
Add Effects:
empacado(?p)
Delete Effects:
etapa(?p, empacar)

Empacar pieza es el cuarto y último proceso y sus precondiciones es que la pieza se encuente en el área de empacar, que este pulida y la máquina de empacar esté libre. Su efecto es que la pieza se defina como empacada. Empacada significa que el proceso de fabricación ha concluido.

4. Estado inicial (Ejemplo).

pieza(p1)
etapa(p1, cortado)
area(p1, área-cortar)
libre(máquina-cortadora)
libre(máquina-taladrar)
libre(máquina-pulir)
libre(máquina-empacar)

5. Objetivo (Goal).

empacado(p1)

6. Secuencia de planificación esperada (Plan).

El plan óptimo STRIPS generado sería:
cortar-pieza(p1)
taladrar-pieza(p1)
pulir-pieza(p1)
empacar-pieza(p1)

Código PDDL del planificador.

```
(define (problem fabricar-pieza-1)
  (:domain piezas-industriales)

  (:objects
    p1 - pieza
    maquina-cortar maquina-taladrar maquina-pulir maquina-empacar - maquina
    area-cortar area-taladrar area-pulir area-empacar - area
  )

  (:init
    ;; Estados iniciales
    (etapa p1 cortar)
    (libre maquina-cortar)
    (libre maquina-taladrar)
    (libre maquina-pulir)
    (libre maquina-empacar)
  )

  (:goal
    (empacar p1)
  )
)


(define (domain piezas-industriales)
  (:requirements :strips :typing)
```

```
(:types pieza area maquina)

;; --------- PREDICADOS ------------------
(:predicates
  (area ?p - pieza ?a - area)
  (etapa ?p - pieza ?e - etapa)
  (cortar ?p - pieza)
  (taladrar ?p - pieza)
  (pulir ?p - pieza)
  (empacar ?p - pieza)
  (libre ?m - maquina)
)

;; --------- CONSTANTES DE ETAPAS ------------
(:constants cortar taladrar pulir empacar - etapa)

;; --------- ACCIONES -------------------------

(:action cortar-pieza
  :parameters (?p - pieza ?m - maquina)
  :precondition (and
    (etapa ?p cortar)
    (libre ?m)
  )
  :effect (and
    (cortar ?p)
    (etapa ?p taladrar)
    (not (etapa ?p cortar))
  )
)

(:action taladrar-pieza
  :parameters (?p - pieza ?m - maquina)
  :precondition (and
    (cortar ?p)
    (etapa ?p taladrar)
    (libre ?m)
  )
  :effect (and
    (taladrar ?p)
    (etapa ?p pulir)
    (not (etapa ?p taladrar))
  )
)

(:action pulir-pieza
  :parameters (?p - pieza ?m - maquina)
  :precondition (and
    (taladrar ?p)
    (etapa ?p pulir)
    (libre ?m)
  )
  :effect (and
    (pulir ?p)
```

```
      (etapa ?p empacar)
      (not (etapa ?p pulir))
    )
  )

  (:action empacar-pieza
    :parameters (?p - pieza ?m - maquina)
    :precondition (and
      (pulir ?p)
      (etapa ?p empacar)
      (libre ?m)
    )
    :effect (and
      (empacar ?p)
      (not (etapa ?p empacar))
    )
  )
)
```

Plan generado.

```
(cortar-pieza p1 maquina-empacar)
(taladrar-pieza p1 maquina-empacar)
(pulir-pieza p1 maquina-empacar)
(empacar-pieza p1 maquina-empacar)
```

Conclusiones.

Es relevante el potencial del uso de PDDL como herramienta para modelar y automatizar procesos industriales complejos. El estudio demuestra que, aun en su etapa preliminar, un planificador basado en PDDL puede representar de manera precisa la lógica operativa de una línea de producción, permitiendo generar planes óptimos y estructurados para el ensamblaje de piezas. La formalización de etapas como cortar, taladrar, pulir y empacar evidencia que los procesos industriales pueden descomponerse en acciones con precondiciones y efectos bien definidos, lo que facilita la validación, el análisis y la futura optimización del flujo de trabajo.

Asimismo, se confirma que las raíces conceptuales provenientes de STRIPS y la evolución del lenguaje PDDL proporcionan un marco robusto para el modelado de dominios industriales reales, incluso cuando existen restricciones de recursos, secuencias obligatorias y condiciones de seguridad. La implementación utilizando herramientas como editor.planning.domains muestra que los planificadores pueden integrarse con relativa facilidad en entornos educativos y de prototipado, permitiendo iteraciones rápidas y mejoras progresivas del modelo.

Créditos.

Los autores agradecen al Tecnológico Nacional de México por el financiamiento del proyecto 21520.25-P “Planificador inteligente de ensamblaje industrial en electromovilidad automotriz implementado en lenguaje de definición de dominios (PDDL)”, de la convocatoria de apoyo a proyectos de desarrollo tecnológico e innovación 2025 y las facilidades del Instituto Tecnológico de Tuxtla Gutiérrez para la realización de este trabajo.

Referencias bibliográficas.

Blum A. y Furst M. (1995). "Graphplan algorithm," 1995.

Education.Planning.Domains (2025). A central source for educational resources for planning domains, http://planning-domains-education.haz.ca/

Fikes y Nilsson (1971). "Strips: A new approach to the application of theorem proving to problem solving," 1971.

Fox, M., & Long, D. (2003). PDDL2.1: An extension of PDDL for expressing temporal planning domains. Journal of Artificial Intelligence Research, 20, 61–124. https://doi.org/10.1613/jair.1129**.**

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated Planning: Theory and Practice*. Morgan Kaufmann. https://api.pageplace.de/preview/DT0400.9780080490519_A25022382/preview-9780080490519_A25022382.pdf.

ICAPS (2025). International Conference on Automated Planning and Scheduling, https://www.icaps-conference.org/.

McDermott, D., et al. (1998). PDDL — The Planning Domain Definition Language (AIPS-98 technical report). (Documento fundacional del lenguaje PDDL). https://www.cs.cmu.edu/~mmv/planning/readings/98aips-PDDL.pdf**.**

McDermott, D. M., (1998). The 1998 AI planning systems competition, AI ma-gazine, 2, pags. 35–55, 2000.

Newell y Simon (1959). "Report on a general problem-solving program," 1959.

Rae (2025). Real Academia Española. https://www.rae.es/.

Información de los autores.

Arturo Fernando Cruz Reyes, Ingeniero en Sistemas Computacionales egresado del Instituto Tecnológico de Tuxtla Gutiérrez, realizó su tesis profesional con el proyecto llamado "Generación de rutas de aprendizaje basado en modelos de planificación de inteligencia artificial".

Mauricio Antonio Guillén Meza es estudiante/profesional de Ingeniería en Sistemas Computacionales en el Instituto Tecnológico de Tuxtla Gutiérrez. Se interesa por la programación, el análisis de sistemas, infraestructura y el desarrollo de soluciones tecnológicas, destacándose por su responsabilidad y aprendizaje constante.

Héctor Guerra Crespo es egresado del I. T. de Mérida (Yucatán, México) de la carrera de Ingeniería en Sistemas Computacionales en 1994, es Doctor en Sistemas Computacionales por la Universidad del Sur (Chiapas, México) en 2011. Es profesor en el área de Ingeniería en Sistemas Computacionales del Instituto Tecnológico de Tuxtla Gutiérrez y en el área de Licenciatura en Sistemas Computacionales de la Universidad Autónoma de Chiapas, en ambas desde 1995.

Luis Tomás García Andrade, Ingeniero en Electrónica por el Instituto Tecnológico de Tuxtla Gutiérrez, Maestro en Administración de Redes y Telecomunicaciones por el Instituto de Estudios Superiores de Chiapas, Docente del Departamento de Eléctrica y Electrónica del Instituto Tecnológico de Tuxtla Gutiérrez.

Néstor Antonio Morales Navarro es Doctor en Desarrollo Tecnológico. Docente del departamento de la carrera de Ingeniería en Sistemas Computacionales y Jefe del departamento del Centro de Cómputo del Tecnológico Nacional de México campus Tuxtla Gutiérrez. Miembro del Sistema Nacional de Investigadores, nivel Candidato. Se especializa en el área de Visión e Inteligencia Artificial.