Article

Not peer-reviewed version

PlanProjU: Automated Planning System for University Projects

Jhon Wilder Sanchez-Obando * , Néstor Dario Duque-Méndez , Luis Fernando Castillo-Ossa

Posted Date: 14 February 2026

doi: 10.20944/preprints202602.1153.v1

Keywords: PlanProjU; automated planning; HTN; SHOP 2; PyHOP; plans; BPMN

 doi:10.20944/preprints202602.1153.v1

Article

PlanProjU: Automated Planning System for University Projects

Jhon Wilder Sanchez-Obando [1,*], Néstor Dario Duque-Méndez [2] and Luis Fernando Castillo-Ossa [3]

[1] University of Caldas
[2] Department of Computer Science and Computing, Universidad Nacional de Colombia
[3] Universidad de Caldas, Departamento de Sistemas e Informática, Manizales, Colombia; Universidad Nacional de Colombia, Departamento de Informática y computación, Manizales, Colombia
* jhon.sanchez@ucaldas.edu.co

Abstract

Effective implementation of university extension projects requires a structured approach with explicit objectives, dependencies, and resource constraints. Automated planning, particularly Hierarchical Task Network (HTN) planning, provides an operational method to schedule complex activities by decomposing high-level goals into executable tasks. This paper presents a tool that converts university project documents into BPMN 2.0 declarative processes and automatically produces HTN planning inputs. Following the Design Science Research methodology, the architecture integrates (i) a pre-planner parser that extracts activities, roles, and precedence relations from BPMN models, (ii) generators that create domain and problem files, and (iii) the HTN planners SHOP 2 and PyHOP to synthesize executable plans. The system was validated with three categories of projects provided by two public universities in Colombia: Universidad Nacional de Colombia and Universidad de Caldas. The platform produces multiple alternative plans for each project and reports plan length and solution-search cost, enabling direct comparison across planners. Results show that the proposed workflow reduces manual scheduling effort, improves consistency of implementation roadmaps, and supports evidence-based selection of implementation strategies under different constraints. These capabilities help extension offices formalize knowledge, audit decisions, and reuse plans across initiatives. Traceability links BPMN elements to planning tasks.

Keywords: PlanProjU; automated planning; HTN; SHOP 2; PyHOP; plans; BPMN

1. Introduction

The design and execution of university projects are commonly carried out following the Project Management Professional (PMP) methodology [1]. However, in the context of official university outreach programs, the PMP methodology is often not fully implemented due to technical and structural limitations in higher education institutions. This gap indicates that having a generic project management methodology is insufficient; instead, there is a pressing need for detailed planning across all project phases.

From an AI perspective, the field of AI planning also known as automated planning provides a well-established computational framework to automate the entire planning process through the generation of executable plans, thereby streamlining the planning process and enhancing efficiency [2,3].

Recent progress in project planning has been driven by significant advances in decision-making, resource allocation, risk analysis, cost-benefit evaluation, and financial assessment techniques [4]. These traditional approaches have consistently proven effective in numerous contexts, yet they remain inherently limited in their ability to facilitate real-time monitoring and the explicit consideration of uncertainty factors that are particularly relevant in university environments. Over

 doi:10.20944/preprints202602.1153.v1

the past several decades, research has extensively explored diverse methodologies, mathematical models, and algorithms aimed at enhancing project planning efficiency [5–9].

A contemporary trend emerging alongside the rise of Generative AI is the integration of large language models (LLMs) into project planning. For instance, Zhen et al. [10] proposed a deep-learning-based task planning system embedded in an LLM to emulate human expert reasoning in task scheduling, demonstrating the ability of LLMs to enhance project planning capabilities. Similarly, Schröder [11] applied knowledge engineering to design a reasoning framework that, when integrated with an LLM, supports engineering project planning under the SCRUM methodology, further highlighting the benefits of LLMs in project planning. However, despite these advances, most applications remain confined to robotics and software engineering, revealing a significant research gap in the application of LLMs to broader project management domains, underscoring the need for further exploration and development in this area.

The integration of large language models (LLMs) into automated planning has been explored in recent attempts, as surveyed by Höller (2020) [12]. However, this approach is hindered by fundamental challenges, including hallucinations, which compromise the reliability of reasoning when transitioning from an initial to a goal state. This limitation is further emphasized by the fact that these challenges undermine the effectiveness of LLM-based project planning methods, highlighting the need for alternative approaches. Automated planning techniques, which generate plans using embedded constraints and heuristics, given a domain description and initial/goal states, offer a viable alternative [13].

One promising subfield is hierarchical task network (HTN) planning, which decomposes complex tasks into sequences of primitive, executable actions [14,15]. While primitive actions are explicitly defined by preconditions and effects, compound tasks lack such explicit semantics in the most widely used formalism, HDDL [15] the domain description language employed in the International Planning Competition (IPC) in 2020 and 2023 [16].

A key challenge in HTN planning is the manual creation of domain models, which demands expert-level knowledge. This process can be supported by structured representations of real-world knowledge, such as the Business Process Model and Notation (BPMN). Rooted in the principles of Frederick Taylor's scientific management [17], BPMN assumes the existence of an underlying fixed business process (BP) that can be automated and executed by a process management scheduling (PMS) system [18,19].

The process of implementing a repeatable business process involves a structured approach that includes three key steps. First, it requires identifying a repeatable business process (BP) a structured abstraction of a real workflow. This step is crucial as it lays the foundation for the subsequent steps. The second step involves formalizing the identified BP into a process model that captures the task execution flows toward a business objective. This process model is often created using the explicit control flow in the Business Process Model and Notation (BPMN), a widely accepted standard for modeling business processes. The use of BPMN enables a clear and unambiguous representation of the process, facilitating effective communication and understanding among stakeholders. The third and final step involves automating the routing of the process via a PMS. The PMS assigns tasks to the appropriate participants, whether human or software agents, ensuring that the process is executed efficiently and effectively. This automation enables a seamless transition from an initial to a goal state, thereby justifying the representation of project management as a BPMN-modeled process.

BPMN offers two distinct advantages for modeling university projects: (1) it provides a graphical, hierarchical representation of cause–effect dependencies, and (2) its XML serialization can be parsed to identify state transitions in each project component. These transitions encode conditional operations for decision flows, yielding sequential and parallel task executions with their respective constraints [20].

Once the BPMN diagram is parsed, two HDDL files are generated: the domain file containing the HTN action model and the problem file defining the initial and goal states of specific university projects. The SHOP2 planner introduced in the 2002 IPC and originally implemented in LISP [2,22]

 doi:10.20944/preprints202602.1153.v1

was then applied. Unlike its predecessor, SHOP2 supports partially ordered subtasks, improving efficiency when operating over simple knowledge bases.

PyHOP is a simple HTN scheduler written in Python, compatible with versions 2.7 through 3.2, and contains fewer than 150 lines of code. Its scheduling algorithm is based on that of SHOP [22], but it avoids the need for a specific scheduling language by having the task network and its methods written in Python, using alternative syntax to logic for defining variables, actions, and methods. PyHOP's development stemmed from the observation that application developers often wrote their own scheduling systems instead of learning specialized AI scheduling languages [23]. PyHOP was created with the hope of providing an HTN scheduler that would be understandable to people without AI experience. The author of Pyhop did not make much of an effort to promote it, but its ease of understanding and use has made it useful for rapid prototyping, leading to its use in various projects and publications.

Prior work combining automated planning and BPMN for process automation has demonstrated the effectiveness of this approach. Schuschel's work (2006) [24] is a notable example, as it utilizes classical planning to automate process definitions for adaptive workflows, showcasing the ability to adapt to changing circumstances. Similarly, Moreno (2005) [25] successfully integrated knowledge models into temporal classical planners for telecommunications network installation, highlighting the effectiveness of this approach in complex domains. Furthermore, Da silva (2017) [26] has addressed the challenges of human knowledge representation by utilizing inductive logic programming to discover action operators in classical planning, underscoring the need for improved knowledge representation in automated planning systems.

However, these contributions are limited to temporal and numeric classical planning without addressing HTN planning, which is a significant gap given the complexity of modern project management. This study addresses three critical gaps: i) the scarcity of automated execution systems for university projects; ii) the absence of HTN-based project automation to produce context-adaptive plans for dynamic project management conditions, which can result in inflexible and ineffective planning; and iii) the lack of project planning systems prioritizing automatic plan generation for university project execution, which can hinder the ability to respond to changing project requirements.

The objectives of this work are as follows: (a) To develop PlanProjU (Planning + Projects + University), a comprehensive HTN-based automated planning system specifically tailored for university projects, thereby enhancing efficiency and productivity in project management. (b) To represent real-world university project knowledge via BPMN diagrams, providing a standardized and visual framework for project planning and execution. (c) To validate PlanProjU through the generation of domains and problem instances, enabling the automatic production of executable project plans, thereby ensuring the accuracy and reliability of the system.

The main contributions of this research are multifaceted and far-reaching, encompassing both technical advancements and practical applications. First, the design and implementation of an innovative HTN-based automated planning framework for university projects represents a significant breakthrough in the field of project management. This framework has the potential to streamline project execution, enhance efficiency, and improve overall outcomes. Furthermore, the extension of HTN planning to non-traditional domains such as beyond robotics, manufacturing, risk management, and logistics demonstrates the versatility and adaptability of this planning approach. By applying HTN planning in these diverse domains, researchers can identify commonalities and develop more generalizable solutions, ultimately contributing to the advancement of the field. Theall, the integration of AI planning with organizational management represents a crucial step toward bridging the gap between theoretical AI research and practical projects. This integration has the potential to provide a more comprehensive understanding of how AI can be effectively used in other real-world contexts.

The remainder of this paper is organized in a logical and coherent manner, allowing for a clear progression of ideas. Specifically, the structure is as follows: Section 2 provides a comprehensive

 doi:10.20944/preprints202602.1153.v1

review of the relevant literature, establishing a foundation for the work presented in this paper. Section 3 outlines the methodology employed in developing PlanProjU, providing a detailed understanding of the approach taken. Section 4 show experiments. Section 5 presents the results of this development, allowing for an evaluation of the effectiveness of the methodology. Section 6 engages in a thorough discussion of the findings, highlighting key insights and implications. Section 7 concludes the paper. Finally, Section 8 shows future work, summarizing the main contributions and outcomes.

2. Related Work and Definitions

This section highlights the significance of research that has investigated the integration of automated planning for BPMN process modeling with other applications designed for constructing HTN planning domains and problems. A total of 27 studies that focused on the creation of domains and problems to generate plans that enable the implementation of process models and the generation of workflows supported by automated planning were identified.

2.1. Exploratory Search Alternative

For several decades, the AI community has been actively involved in business process management research, with a significant body of work examining the role of automated planning technologies in process management systems. These systems are designed to manage complex business processes while maintaining robustness, reactivity, and adaptability in response to environmental and tasking changes. One of the initial works addressing this research challenge is reference Reijers (2016) [27], which outlines how planning techniques from the planning community can be utilized to synthesize new business processes and repair previously defined processes that are no longer suitable for a given situation. This highlights the potential of planning techniques to improve business processes' flexibility and responsiveness. Marella (2019) [28] further investigates the use of an intelligent assistant based on planning techniques, which may suggest compensation procedures or re-execution of activities if anticipated failures occur during process execution. This demonstrates the ability to plan techniques to mitigate potential failures and ensure the smooth operation of business processes. In M. Rosa (2017) [29] describe how planning can be integrated with process execution and plan refinement and investigate plans patching and plan repair as means to enhance flexibility and responsiveness. This integration of planning with process execution is crucial for ensuring that business processes remain adaptable and responsive to changing circumstances.

2.2. Automated Planning for BPMN

The application of planning techniques is also relevant in the field of web service composition, where complex challenges are addressed using planning. Notably, a limited number of studies employ domain-independent planners to generate compositions of atomic web services, demonstrating the effectiveness of planning in this area. Research in [28–31] has explored the use of planning technologies to achieve service composition at the knowledge level, as seen in [34], highlighting the potential of planning to address complex web service composition challenges. Furthermore, studies such as Hoffman [35] have utilized customized planners, including the FF planner, to construct service orchestrations from atomic IT entities described in a planning-like manner, showcasing the versatility of planning techniques in web service composition. Additionally, research in [34,35] has combined situation calculus with planning techniques to perform service composition, providing a comprehensive approach to addressing web service composition challenges. In contrast to works that view services as atomic planning operators, a significant body of research [36,37] has considered services as stateful entities, characterized by a well-defined behavioral description derived from BPEL specifications. In these studies, the requirements of the desired composite service are expressed in a temporal logic-like language, and planning techniques based on model-checking are employed for service composition, underscoring the importance of

 doi:10.20944/preprints202602.1153.v1

considering the stateful nature of services in planning-based web service composition. Similarly, recent advances in symbolic search such as the work of Bertoli (2006) [39] indicate that complex scenarios such as process modeling require the partitioning of action models. On the other hand, the development of research in process modeling has been developed mainly from the classical AP approach and not from HTN, due to the limitation of a development framework in HDDL. Given this difficulty Berchner (2025) [40]has developed a framework for HDDL like the development frameworks in PDDL to improve the construction of action and problem models in alternative contexts such as process modeling.

2.3. Contribution and Novelty

PlanProjU diverges from existing research by treating project planning as a collection of distinct, dynamic components without outlining the comprehensive transformation process from BPMN to an HTN planning domain and problem. A novel knowledge-representation framework is proposed, which extracts control-flow and data semantics from arbitrary BPMN structures and compiles them into two primary components: (i) action models, comprising tasks/operators with preconditions, effects, and resources, and (ii) a planning problem with ordering constraints derived from gateways, events, and artifacts. This results in a pipeline that generates executable HTN plans, eliminating the need for intermediate Gantt encodings, while maintaining a direct connection to the original process specification and organizational rules.

In contrast to earlier surveys of AI planning for process control flows, such as Stein (2008) [41] which fails to address implementable project-management instantiation, our approach takes a more comprehensive approach by treating the BPMN project model itself as the canonical source, then translating it into an HTN domain/problem for plan synthesis. This approach differs from Vanhatalo (2009) [42] propose knowledge-representation frameworks to aid experts in crafting action models, but do not explicitly ground real-world objects and resources within action schemas. Similarly, Barták (2010) [43] consider automatic action-model generation from elicited expert knowledge, but do not encode organizational knowledge in BPMN-like process frames as the substrate for constructing planning domains and problems. In contrast, PlanProjU closes these gaps by coupling BPMN-level semantics with HTN compilation, enabling explainable, policy-aware project plans that are directly traceable to institutional processes. Köckemann (2025) [44] emphasizes the importance of promoting the use of AI Planning across various domains, yet this goal remains unfulfilled due to the significant challenge of developing action and problem models by experts in AI Planning. Notably, the proposed AIDDL development framework has addressed the creation of action models for traditional and temporal planning, but it does not extend to HTN planning, thereby leaving a notable gap in its scope.

2.4. HTN Definitions

An HTN planning instance can be written as $P = \langle L, C, A, M, s_0, t_{nI}, g, \delta \rangle$. Here, L, A, s_0, g and δ have their usual meanings from classical planning (set of literals, set of primitive actions, initial state, goal condition, and cost/metric, respectively). The set C contains abstract task symbols (high-level activities) and is disjointed from the primitive task names, i.e., $C \cap A = \emptyset$. Tasks are arranged into task networks $tn = \langle T, \prec, \alpha \rangle$, where T is a (possibly empty) set of task identifiers, and $\alpha: T \to A \cup C$ labels each identifier with either a primitive or an abstract task—allowing the same task name to occur multiple times in the network. The precedence relation $\prec \subseteq TxT$ is a strict partial order over identifiers (irreflexible, transitive, and therefore asymmetric). The component M denotes the set of methods that decompose abstract tasks into subnetworks, and t_{nI} is the initial task network.

Two task networks $tn = \langle T, \prec, \alpha \rangle$ and $tn' = \langle T', \prec', \alpha' \rangle$ are isomorphic when they differ only by a renaming of task identifiers. Formally, there exist a bijection $\sigma: T \to T'$ such that for all $t, t' \in T$: $(t, t') \in \prec \Leftrightarrow \big(\sigma(t), \sigma(t')\big) \in \prec'$ and $\alpha(t) = \alpha'(\sigma(t))$.

In HTN planning, decompositions are specified by methods. A method couples an abstract task symbol $c \in C$ with a subtask network, written $m = \langle c, u \rangle$, where $u = \langle U, \prec_U, \alpha_U \rangle$. Consider a task network $tn_1 = \langle T_1, \prec_1, \alpha_1 \rangle$ and an identifier $t \in T_1$ such that $\alpha_1(T) = c$. The method m can be

 doi:10.20944/preprints202602.1153.v1

applied by replacing t with a fresh copy of the subtask network into a task network $tn_2 = \langle T_2, \prec_2, \alpha_2 \rangle$. Formally, there must exist an isomorphic renaming θ of u that produces $\hat{u} = \langle \hat{U}, \prec_{\hat{U}}, \alpha_{\hat{U}} \rangle$ with fresh identifiers $(T_1 \cap \hat{U} = \emptyset)$ and $u \cong \hat{u}$. There is a task network $tn' = \langle T', \prec', \alpha' \rangle$ with $tn' \cong tn$ and $T_1 \cap T' = \emptyset$. The task network tn_2 is defined as follows:

$$tn_2 = ((T_1 \backslash \{t\}) \cup T', \prec' \cup \prec_D, (\alpha_1 \backslash \{t \to c\} \cup \alpha')), \tag{1}$$

$$\begin{aligned} \prec_D = & \{(t_1, t_2) | (t_1, t) \in \prec_1, t_2 \in T'\} \\ & \cup \{(t_1, t_2) | (t, t_2) \in \prec_1, t_1 \in T'\} \\ & \cup \{(t_1, t_2) | (t_1, t_2) \in \prec_1, t_1 \neq t \wedge t_2 \neq t\} \end{aligned} \tag{2}$$

M is the set of decomposition methods. We will write $tn_{t,m} \to tn'$ to denote that tn can be decomposed into a task network tn' by applying the method m to the task t, and $tn \to* tn'$ if it is possible to decompose tn into tn' using a sequence of methods.

The decomposition begins either with a single initial task or with a task network (that may include more than one task). A definition using a single initial task is sometimes beneficial to keep proof simple. Using an initial task network makes it possible to interpret every search node in the search space as a new planning problem [45]. However, both definitions are equivalent: Given an initial task network tn_I, it can be compiled away by introducing a new task c_I (that is the new initial task) and a method m that decomposes the new task into the original task network $m = (c_I, tn_I)$.

A task network $tn = (T, \prec, \alpha)$ is a solution to a given HTN planning problem $P = (L, C, A, M, s_0, tn_I, g, \delta)$ if and only if the following solution criteria hold:

- $tn_I \to* tn$, i.e. It can be reached by decomposing the initial task network.
- $\forall t \in T: \alpha(t) \in A$, i.e. all task names are primitive.
- There is a sequence $\langle t_1, t_2 \ldots . t_n \rangle$ of the task in T that is in line with $\prec$, i.e. $\forall i < j: (t_j, t_i) \notin \prec$, and the application of $\langle \propto (t_1) \propto (t_2) \ldots . . \propto (t_n) \rangle$ in s_0 results in goal state.

The last solution criterion requires an HTN solution to result in a goal state. This is quite unusual, since most hierarchical formalisms lack a (state-based) goal definition though some have one, e.g. these introduced by Bercher (2017) [46] and [47].

2.5. SHOP 2

SHOP 2 was first demonstrated in 2002 at the "International Planning Competition" [48]. It was originally written in Lisp, a functional programming language like SHOP. One of SHOP 2's key properties is that it maintains the same order of planning tasks as they will be executed, which enables it to generate plans quickly [21]. However, SHOP 2 differs from SHOP in one significant way: it allows subtasks to be partially ordered, while still preserving performance. This modification also enables the use of simpler knowledge bases. Furthermore, SHOP 2 can interleave subtasks from different tasks using the Planning Operator Transfer and Decomposition (POTD) method [1] and [21], thereby avoiding the need for global planning instructions.

As explained in reference [21], the basic elements of SHOP 2 are as follows: Logical Atoms, which consist of a predicate name combined with a list of arguments, World State, which is a collection of ground Logical Atoms, and Tasks, which are syntactically similar to Logical Atoms but differ semantically, consisting of a task name and a list of arguments. In SHOP and SHOP 2, domain-specific knowledge is necessary for planning within a particular domain. This knowledge includes Axioms, which are expressed using Horn-Clause, Methods, which function as if-then-else statements, and Operators, which comprise a Task Atom, a Delete List, and an Add List. The Operators modify atoms within the World State. Additionally, Operators can include "protection request" and "protection cancelation," indicating that certain conditions should not be removed or should be considered removable, respectively. In our case, we utilize "Effects," which can be used to modify global variables, mimicking the functionality of the Delete and Add lists.

Task: The task is an activity to be performed, which can be either a Primitive task or a Compound task. Primitive tasks utilize their task symbol as the Operator name and the Task arguments as the Operator parameters. If the task is Compound, it must be decomposed into subtasks

 doi:10.20944/preprints202602.1153.v1

using Methods. Operator: An Operator represents a Primitive task to be performed. It consists of a head, which includes the name and arguments of the task, Preconditions expressions that must be satisfied, and an Effects list, which replaces the Add and Delete lists. No two operators can have the same name, and all applications of an Operator are instances of the same Operator. Each Operator can have a "cost" expression; the cost of all plans is the sum of the operator's costs. Method: A Method indicates how to decompose a compound task into a subset of tasks. It consists of a head, which includes the name of the task, a list of Preconditions that must be satisfied, and a list of sub-tasks to be decomposed into. SHOP 2 allows the use of the keyword ": ordered" to force sub-tasks to be Totally Ordered. If not used, the subtasks can be Partially Ordered, or by using the keyword ": unordered". The method can have a form: $(:method\ head\ (m)\ p_1\ t_1\ p_2\ \ldots\ldots p_n t_n)$ Decomposition can be viewed as an if-then-else statement. The first requirement, p1, is checked. If it is satisfied, the subtasks t1 are applied. If not, the planner moves on to the next requirement, p2, and repeats this process until a requirement is satisfied. This can result in different ways to solve the problem, generating alternative branches in the search space.

To start the planner SHOP 2 it is required to define: $s: initial\ state, T: partially\ ordered\ set\ of\ sub\ tasks\ and\ D: domain\ or\ action\ model$. Starting with a: run SHOP 2(S, t, D). The Pseudo-code example from [1]:

$$M \leftarrow m, \theta$$
$$if\ M = 0\ then\ return\ failure$$
$$non - deterministically\ choose\ a\ pair\ (m, \theta) \in M$$
$$modifiy\ T\ by\ removing\ t, adding\ sub(m)$$
$$if\ sub(m)\ != 0\ then$$
$$T_0 \leftarrow t \in sub(m)$$
$$else\ T_0 \leftarrow t \in T$$

SHOP 2 is considered more beneficial for real-world domains than for classical planning operators because it more accurately represents the domain designer's perspective on the problem. Although SHOP 2 features a domain-independent planning engine, its HTN methods are domain-specific and must be modified for each individual problem. The incorporation of domain-specific problem-solving knowledge can enhance a planner's performance.

2.6. HDDL

The Hierarchical Domain Definition Language (HDDL) is an extension of the Planning Domain Definition Language (PDDL) that incorporates hierarchical task networks (HTN). Introduced by Holler [15], HDDL offers a standardized language for hierarchical planning systems, accompanied by comprehensive documentation and a range of domains and problems. Many of these resources are sourced from the hierarchical task network tracks of the International Planning Competitions (IPC-HTN), specifically from the 2023 IPC-HTN tracks. HDDL's design enables users to define or modify problem-solving approaches by adjusting the hierarchical task networks to suit their specific needs.

Definition of planning action model or Domain: A planning domain D is tuple (L, T_p, T_c, M) it define: L is the underlying predicate logic, $T_p\ and\ T_c$ are finite sets of primitive and compound tasks, M is a finite set of decomposition methos with compound tasks from T_c and tasks networks over the set $T_p \cup T_c$.

Definition of planning problem: A planning problem P is a tuple (D, s_I, tn_I, g), it defines: $s_I \in S$ is the initial state, a ground conjunction of positive literals over the predicates assuming the closed word assumption. tn_I is the initial task network that may not necessarily be grounded. g is the goal description, being a first-order formula over the predicates (not necessarily ground).

In other words, beyond the action definition in PDDL, which establishes the rules of interaction with the environment, HDDL introduces two additional operators: task and method. The introduction of these operators enhances the expressiveness of HDDL, allowing for a more detailed representation of complex problem-solving scenarios. A task in HDDL represents a high-level action,

 doi:10.20944/preprints202602.1153.v1

which can be broken down into more specific and manageable components. Conversely, a method is a strategy to accomplish a task, indicating that multiple methods can exist to perform a single task. This flexibility is crucial in problem-solving, as it enables the exploration of different approaches to achieve a common goal. Furthermore, a method in HDDL is essentially a task network that decomposes a high-level task into a partially or totally ordered list of tasks and actions, providing a structured framework for planning and execution.

An example of tasks and methods definition in HDDL:

```
(: task get-to: parameters (?l - location))
(: method m-drive-to-via
        : parameters (? li? ld - location)
        : task (get-to? ld)
        : precondition ()
        : subtasks (and
                t1 (get-to? li))
                 t2 (drive? li? ld)))
        : ordering (and
                  (t1 < t2)))
```

An example of the goal in a transport problem is:

```
(: htn
      : tasks (and
            (deliver package-0 city-loc0)
            (deliver package-1 city-loc2))
      : ordering ())
```

2.7. PyHOP

PyHOP is a simple HTN scheduler written in Python, compatible with versions 2.7 through 3.2, and contains fewer than 150 lines of code. Its scheduling algorithm is based on that of SHOP [22], but it avoids the need for a specific scheduling language by having the task network and its methods written in Python, using alternative syntax to logic for defining variables, actions, and methods. PyHOP's development stemmed from the observation that application developers often wrote their own scheduling systems instead of learning specialized AI scheduling languages [23]. PyHOP was created with the hope of providing an HTN scheduler that would be understandable to people without AI experience. The author of Pyhop did not make much of an effort to promote it, but its ease of understanding and use has made it useful for rapid prototyping, leading to its use in various projects and publications.

It typically uses a list of tasks, actions, methods, and goals expressed as functions, which facilitates the decomposition of tasks into task methods and goals into goal methods. Therefore, the PyHOP scheduler can return either the list of tasks or the list of goals. One limitation is that loading different domains and scheduling problems requires restarting Python to re-execute, as it has a limited ability to recognize domain and problem files. However, the domain and problem files can be in JSON, PDDL, or even HDDL format. Another limitation with more advanced schedulers like PyHOP 2.0 and GTPyHOP is the less extensive standard documentation compared to other HTN schedulers.

3. Methodology

This research employs the Design Science Research (DSR) approach, a systematic methodology that aims to develop innovative solutions to real-world problems [49]. DSR serves as a framework for managing project risks, constructing theories, and publishing results. Its relevance in this study lies in its focus on designing artifacts that serve human purposes, ultimately generating scientific knowledge to address real-world issues and contribute significantly to the field. The article seeks to develop implementation plans for university projects using automated planning, a method validated

 doi:10.20944/preprints202602.1153.v1

through case studies [50]. To achieve this objective, two key questions will be addressed: What is the most suitable methodology for building a planning system using automated planning? And how can the methodology be validated to generate plans for the implementation of university projects? The resulting planning system will enable the identification of project implementation plans in universities and facilitate the creation of a general project planning environment applicable across various contexts. The next Figure 1 shows the DSR methodology

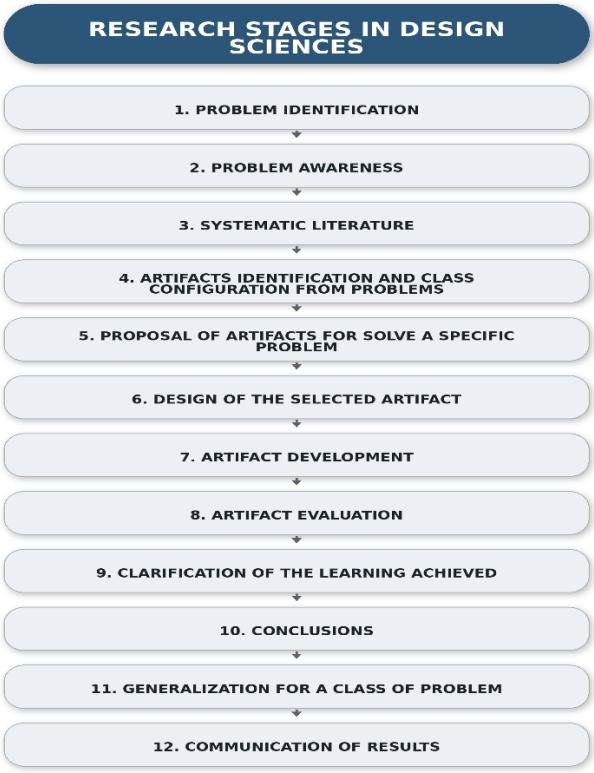

Figure 1. Stages of DSR. Source [50].

Figure 1. The research process in design sciences is a structured and systematic approach that involves several stages to ensure the development of a well-informed and effective solution to a relevant problem. The process commences with the identification of a relevant problem in stage 1, which is essential for formulating a structured research question in stage 2. Understanding the problem and its context, including causes and functionalities of the artifact, is crucial in this stage as it provides a solid foundation for the subsequent stages.

A systematic literature review in stage 3 involves consulting various technical databases, such as Scopus, WoS, IEEE, Google Scholar, Science Direct, and ArXiv, to justify the importance of developing the artifact and solving the problem. This stage is critical as it enables researchers to identify existing knowledge gaps and areas for improvement, thereby informing the development of the artifact. In stage 4, identifying artifacts that address similar problems allows researchers to apply best practices, while configuring the problem class defines the scope of their contributions.

This stage is essential as it enables researchers to build upon existing knowledge and develop a solution that is tailored to the specific problem at hand. The proposal of artifacts in stage 5 is related to solving a specific problem, and the researcher considers the feasibility and reality of the proposed artifacts. This stage is critical as it enables researchers to develop a solution that is practical and effective. In stage 6, the design of the selected artifact involves considering the entire context in which it operates and satisfactory solutions to the study problem.

 doi:10.20944/preprints202602.1153.v1

It is essential to describe all procedures for the construction and evaluation of the artifact in this stage as it provides a clear understanding of the research methodology. The construction of the artifact is carried out in stage 7, while stage 8 involves evaluating the behavior of the artifact to provide a satisfactory solution to the problem. This stage is critical as it enables researchers to assess the effectiveness of their solution and identify areas for improvement. Stage 9 involves clarifying the learning achieved, explaining the factors that contributed to the success of the research and the elements that failed.

This stage is essential as it enables researchers to reflect on their experience and identify areas for improvement. The conclusions in stage 10 present the results of the research, the decisions made during its implementation, and the limitations of the research that may give rise to future studies. This stage is critical as it enables researchers to communicate their findings and contribute to the advancement of knowledge in the field. Stage 11 allows for the generalization of knowledge to similar situations by other organizations, while stage 12 involves communicating the results, contributing to a significant advancement of knowledge through publications in journals, seminars, conferences, and other platforms. This stage is essential as it enables researchers to disseminate their findings and contribute to the broader academic community.

3.1. Design Artefact: Mapping to a Graph Model BPMN

In this step, stage 5 of the DSR methodology is presented, given that the previous stages have been thoroughly developed and validated. The foundation established in the preceding stages provides a robust framework for the subsequent stages, ensuring a cohesive and systematic approach to the research. In this stage, algorithm 1 is introduced, specifically designed for parsing the BPMN diagram. This diagram was previously developed by a process design expert for university outreach projects as a case study, providing a real-world context for the research.

3.2. Algorithm Parser or Translate BPMN to HDDL

The algorithm is designed to automate the process of translating BPMN diagrams into a more structured and machine-readable format, specifically an XML structure. This translation enables the BPMN diagram to be further processed and converted into an HTN planning domain structure in HDDL language, which is a standardized format for representing planning domains and problems. This algorithm facilitates the conversion of BPMN elements, such as activities, gateways, and events, into their corresponding representations in the HTN planning domain structure, including primitive tasks, methods, sub methods, and operators.

3.3. Artifact: PlanProjU

The PlanProjU system, designed within the framework of the DSR methodology, offers a structured approach to project planning. By taking the BPMN diagram of relevant university projects as input, the system translates this information into a HDDL domain, allowing for a comprehensive representation of project details. This process is facilitated by the planning expert, who translates project management descriptions into problems that can be solved by the system. The use of a BPMN diagram as input enables the capture of a significant amount of project semantics, ensuring that the generated plans are accurate and effective. The subsequent application of the SHOP 2 planner generates valid plans for project implementation, thereby providing a systematic and reliable approach to project management. Figure 2 shows the architecture.

 doi:10.20944/preprints202602.1153.v1

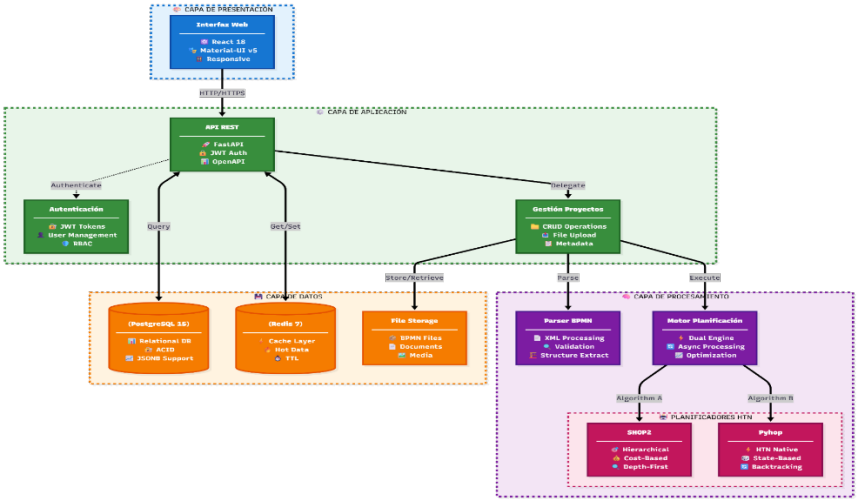

Figure 2. PlanProjU system architecture. Source: self-made.

Figure 2. Shows summarize the layered software architecture of PlanProjU, emphasizing a clear separation of concerns across four tiers and the end-to-end data flow required to generate HTN plans from BPMN models. At the presentation layer, a responsive web interface (React + Material UI) communicates over HTTP/HTTPS with the application layer, which exposes a REST API implemented with FastAPI and documented via OpenAPI, thereby enabling standardized service interaction and traceability. Security and governance are handled through an authentication module based on JWT tokens, user management, and RBAC, which constrains access to project operations and enforces role-dependent capabilities. The project management component orchestrates CRUD operations, file uploads, and metadata handling, delegating persistence responsibilities to the data layer, where PostgreSQL supports transactional storage (ACID) and JSONB structures, Redis provides coaching for hot data and TTL-based performance optimization, and file storage maintains BPMN files and associated documentation. Finally, the processing layer operationalizes the core research contribution: a BPMN parser performs XML processing, structural validation, and model feature extraction, feeding a planning engine that executes planning requests using alternative algorithms (SHOP 2 and Pyhop) to produce comparable plan variants. This pipeline illustrates an architecture designed for modularity and extensibility allowing independent evolution of the BPMN parsing subsystem, the planning back end, and the user-facing interface while also enabling systematic comparison of planning outcomes using consistent inputs and integrated execution within a single platform. Figure 3. Shows the component of PlanProjU.

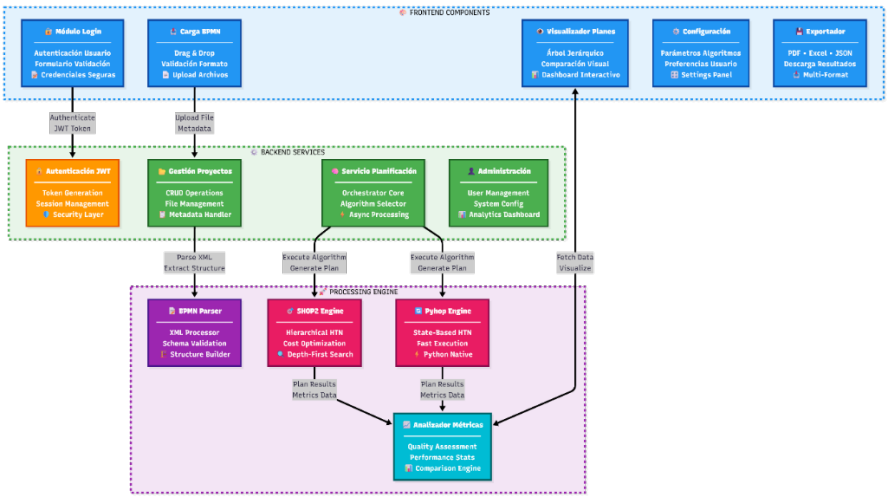

Figure 3. PlanProjU principal components. Source: self-made.

 doi:10.20944/preprints202602.1153.v1

Figure 3. Provides a component-level view of the PlanProjU platform, detailing how the user-facing modules are coupled to backend services and, ultimately, to the HTN planning engine through a controlled processing pipeline. At the **frontend**, the workflow begins with the *Login Module* (secure credentials and validation form), continues with *BPMN Upload* (drag-and-drop plus format validation), and culminates in three key capabilities: (i) a *Plan Visualizer* that supports hierarchical tree inspection and visual comparison, (ii) a *Configuration* panel to set algorithm parameters and user preferences, and (iii) an *Exporter* that generates multi-format outputs (PDF/Excel/JSON). These interactions are mediated by backend services organized around JWT-based authentication (token generation, session management, security layer), project management (CRUD operations, file/metadata handling), a planning service that acts as an orchestrator (algorithm selection and asynchronous execution), and an administration module for user/system configuration and analytics. Within the **processing engine**, the BPMN parser performs XML parsing, schema validation, and structural extraction, producing the planning-ready representation that is then executed by two alternative planners SHOP2 (hierarchical HTN, depth-first search, cost optimization) and a Pyhop-based engine (state-based HTN execution). Plan outputs and runtime metrics are consolidated by a dedicated Metrics Analyzer that supports quality assessment, performance statistics, and comparative evaluation, and the resulting data is fed back to the frontend for visualization and decision support. Overall, architecture evidences a modular design that makes the planning subsystem, metric evaluation, and UI-layer comparison tightly integrated but independently evolve an important property for replicable experimentation and iterative enhancement in a research-grade planning platform. Figure 4 shows the usage layers.

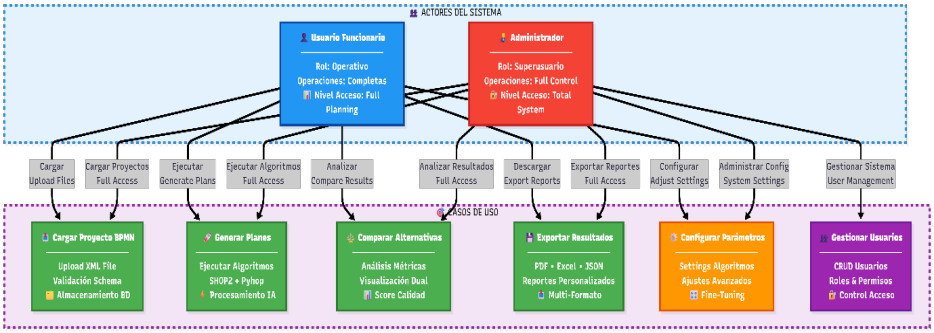

Figure 4. PlanProjU usage layers. Source: self-made.

Figure 4 formalizes the interaction model of PlanProjU by mapping *system actors* to the platform's *core use cases*, making explicit how permissions and responsibilities are operationalized across the workflow. At the top, two actor profiles structure the access-control logic: the Operational User, who executes the end-to-end project planning flow with full access to planning functions, and the Administrator, who holds supervisory privileges with total system access for governance, configuration, and user management. These roles connect to a set of use cases that represent the platform's functional backbone. The operational flow begins with Upload BPMN Project (XML upload, schema validation, and storage), proceeds to Generate Plans (execution of SHOP 2 and Pyhop with AI-driven processing), and then to Compare Alternatives, where users inspect plan outputs through metric-based analysis, visualization, and a consolidated quality score. The workflow is complemented by Export Results, enabling multi-format reporting (PDF/Excel/JSON) and customizable report generation, which supports traceability and offline review. In parallel, the platform exposes governance-oriented functions Configure Parameters (algorithm settings, advanced adjustments, and fine-tuning) and Manage Users (CRUD operations, roles/permissions, and access control) which are primarily aligned with the administrator's responsibilities but may be visible depending on privilege assignment. Overall, the diagram demonstrates a clear separation of concerns between operational execution and system administration, while ensuring that the full

 doi:10.20944/preprints202602.1153.v1

planning lifecycle from BPMN ingestion to plan evaluation and reporting is supported under a coherent role-based access control (RBAC) scheme.

4. Experiments

Experiments were conducted using PlanProjU with a dual objective. Firstly, to verify that well-structured process models selected for this purpose possess a corresponding HTN representation capable of capturing the knowledge inherent in the process. This objective is crucial as it ensures that the HTN representation accurately reflects the complexities of the process, thereby enabling effective planning and scheduling. Secondly, to demonstrate the feasibility of using this approach to facilitate a planning process that generates a contextual plan, considering the ordering of tasks and resource requirements for university projects. The experiments were specifically designed to showcase the planning and scheduling capabilities of the developed approach, thereby providing a comprehensive evaluation of its effectiveness. This demonstrates that knowledge structured in BPMN format can be effectively captured in an HTN action model for planning purposes, given the level of detail typically required for university projects. The successful implementation of this approach highlights its potential for real-world applications, particularly in the context of university projects where complex planning and scheduling are essential.

Specifically, the following are some of the expected results of these experiments: 1) verifying that a corresponding HTN representation exists for process models that include a combination of workflow patterns; 2) finding a plan instance that preserves the semantics associated with those workflow patterns; 3) verifying that the interpretation of the same HTN action model can find different plan instances for different combinations of input parameters. The successful application of PlanProjU in real-world processes demonstrates its practical utility. Notably, it has been utilized in three distinct university projects: patenting, co-creation, and the construction of a science, technology, and innovation laboratory. These projects, managed simultaneously within the framework of open innovation and intellectual capital of two Colombian universities, showcase the versatility of PlanProjU.

Researchers are strongly advised to utilize a computer equipped with a minimum of 32 GB of RAM, a hard drive with a storage capacity of at least 500 GB, and the Python programming language installed. Furthermore, it is highly recommended that they conduct tests on Google Collaboratory from a Gmail account to guarantee compatibility with the user interface. User experience evaluations have consistently shown that the system requires approximately 0.5 minutes to start up. On average, decision makers utilized the BPMN upload function 14.8 times and accessed and saved plan information 15.3 times. In addition, this study systematically tested all execution plan generation processes contextualized with the university project. The plan generation process for different projects was repeated five times, introducing different BPMN models and action enhancements to the HTN planning domain action model. This rigorous validation process helped verify various metrics such as the number of actions, tree depth, length, and size of the plans generated for each project.

The evaluation methodology for the PlanProjU system adhered strictly to the human evaluation approach outlined by Owczarzak (2012) [51], which emphasizes the necessity of establishing explicit evaluation criteria to enable users to objectively assess a system's quality and interoperability. This approach ensures that the evaluation process is thorough and reliable. The system was rigorously tested for one week by eight project managers from diverse fields at Colombian universities, thereby providing a comprehensive and unbiased usability assessment for university project management. For the first experiment, the UNAL-Audifarma extension project was used, which generated the following planning domain and problems in PlanProjU.

Domain:

```
(define (domain university_extension)
(:requirements :typing :hierarchy :equality)
(:types actor resource project material document task)
```

 doi:10.20944/preprints202602.1153.v1


```
(:predicates
      (done ?t - task)
      (available ?r - resource)
      (assigned ?a - actor ?p - project)
      (created ?m - material)
      (approved ?d - document)
      (completed ?t - task)
      (executed ?t - task)
)
(:action solicitar_curso
      :parameters (?a - actor ?r - resource ?p - project)
      :precondition (and (available ?r))
      :effect (and (done ?p) (executed ?p))
)
(:action enviar_propuesta
      :parameters (?a - actor ?r - resource ?p - project)
      :precondition (and (available ?r))
      :effect (and (done ?p) (executed ?p))
)
(:action revisar_y_decidir
      :parameters (?a - actor ?r - resource ?p - project)
      :precondition (and (available ?r))
      :effect (and (done ?p) (executed ?p))
)
(:action definir_fechas_y_horarios
      :parameters (?a - actor ?r - resource ?p - project)
      :precondition (and (available ?r))
      :effect (and (done ?p) (executed ?p))
)
(:action asignar_costos_adicionales_empresa_asume
      :parameters (?a - actor ?r - resource ?p - project)
      :precondition (and (available ?r))
      :effect (and (done ?p) (executed ?p))
)
(:action asignar_docente_y_plataforma_meet
      :parameters (?a - actor ?r - resource ?p - project)
      :precondition (and (available ?r))
      :effect (and (done ?p) (executed ?p))
)
(:action mdulo_1_generalidades
      :parameters (?a - actor ?r - resource ?p - project)
      :precondition (and (available ?r))
      :effect (and (done ?p) (executed ?p))
)
(:action mdulo_2_habilidades_de_un_buen_formador
      :parameters (?a - actor ?r - resource ?p - project)
      :precondition (and (available ?r))
      :effect (and (done ?p) (executed ?p))
)……
```

Problem 1

```
(define (problem docente_problem)
```

 doi:10.20944/preprints202602.1153.v1

(:domain university_extension)
(:objects
a_docente - actor
r_docente - resource
p_docente - project
m_docente - material
d_docente - document
)
(:init (available r_docente))
(:tasks (t1 (docente_project a_docente r_docente p_docente m_docente d_docente)))
(:goal (done p_docente))
)

Problem 2

(define (problem universidad_nacional_de_colombia_problem)
(:domain university_extension)
(:objects
a_universidad_nacional_de_colombia - actor
r_universidad_nacional_de_colombia - resource
p_universidad_nacional_de_colombia - project
m_universidad_nacional_de_colombia - material
d_universidad_nacional_de_colombia - document
)
(:init (available r_universidad_nacional_de_colombia))
(:tasks (t1 (universidad_nacional_de_colombia_project a_universidad_nacional_de_colombia r_universidad_nacional_de_colombia p_universidad_nacional_de_colombia m_universidad_nacional_de_colombia d_universidad_nacional_de_colombia)))
(:goal (done p_universidad_nacional_de_colombia))
)

Problem 3

(define (problem universidad_nacional_de_colombia_problem)
(:domain university_extension)
(:objects
a_universidad_nacional_de_colombia - actor
r_universidad_nacional_de_colombia - resource
p_universidad_nacional_de_colombia - project
m_universidad_nacional_de_colombia - material
d_universidad_nacional_de_colombia - document
)
(:init (available r_universidad_nacional_de_colombia))
(:tasks (t1 (universidad_nacional_de_colombia_project a_universidad_nacional_de_colombia r_universidad_nacional_de_colombia p_universidad_nacional_de_colombia m_universidad_nacional_de_colombia d_universidad_nacional_de_colombia)))
(:goal (done p_universidad_nacional_de_colombia))

For the second experiment, the extension project of the University of Caldas #304 was used, which generated the following planning domain and problem in PlanProjU.

Domain:

(define (domain university_extension)
(:requirements :typing :hierarchy :equality)
(:types actor resource project material document task)
(:predicates
(done ?t - task)

 doi:10.20944/preprints202602.1153.v1

(available ?r - resource)
(assigned ?a - actor ?p - project)
(created ?m - material)
(approved ?d - document)
(completed ?t - task)
(executed ?t - task)
)
(:action recopilar_informacin_del_proyecto
:parameters (?a - actor ?r - resource ?p - project)
:precondition (and (available ?r))
:effect (and (done ?p) (executed ?p))
)
(:action diligenciar_datos_generales_ttulo_modalidad_coordinador_duracin_fechas_lugar
:parameters (?a - actor ?r - resource ?p - project)
:precondition (and (available ?r))
:effect (and (done ?p) (executed ?p))
)
(:action redactar_descripcin_problema_y_antecedentes
:parameters (?a - actor ?r - resource ?p - project)
:precondition (and (available ?r))
:effect (and (done ?p) (executed ?p))
)…….
Problem:
(define (problem proyecto_principal_problem)
(:domain university_extension)
(:objects
t_recopilar_informacin_del_proyecto - task
t_diligenciar_datos_generales_ttulo_modalidad_coordinador_duracin_fechas_lugar - task
t_redactar_descripcin_problema_y_antecedentes - task
t_definir_articulacin_con_planes_institucionales_pdi_ppu_plan_facultaddepartamento - task
t_definir_objetivo_general_y_objetivos_especficos - task
t_definir_metodologa_diseo_desarrollo_evaluacin_y_oferta - task
t_definir_catlogoportafolio_de_educacin_continuada_cursos_diplomados_talleres - task
t_definir_poblacin_beneficiada_y_cantidad_esperada - task
t_definir_novedad_de_la_propuesta - task
t_definir_impacto_esperado - task
t_definir_productos_esperados - task
t_revisar_coherencia_y_completar_la_propuesta - task
t_ajustar_contenido_y_anexos_de_la_propuesta - task
t_enviar_propuesta_a_la_convocatoria_interna - task
proyecto_principal - actor
)
(:init
(pending t_recopilar_informacin_del_proyecto)
(assigned_to t_recopilar_informacin_del_proyecto proyecto_principal)
(can_start t_recopilar_informacin_del_proyecto)
(pending
t_diligenciar_datos_generales_ttulo_modalidad_coordinador_duracin_fechas_lugar)

 doi:10.20944/preprints202602.1153.v1

(assigned_to t_diligenciar_datos_generales_ttulo_modalidad_coordinador_duracin_fechas_lugar proyecto_principal)
(pending t_redactar_descripcin_problema_y_antecedentes)
(assigned_to t_redactar_descripcin_problema_y_antecedentes proyecto_principal)
(pending t_definir_articulacin_con_planes_institucionales_pdi_ppu_plan_facultaddepartamento)
(assigned_to t_definir_articulacin_con_planes_institucionales_pdi_ppu_plan_facultaddepartamento proyecto_principal)
(pending t_definir_objetivo_general_y_objetivos_especficos)
(assigned_to t_definir_objetivo_general_y_objetivos_especficos proyecto_principal)
(pending t_definir_metodologa_diseo_desarrollo_evaluacin_y_oferta)
(assigned_to t_definir_metodologa_diseo_desarrollo_evaluacin_y_oferta proyecto_principal)
(pending t_definir_catlogoportafolio_de_educacin_continuada_cursos_diplomados_talleres)
(assigned_to t_definir_catlogoportafolio_de_educacin_continuada_cursos_diplomados_talleres proyecto_principal)
(pending t_definir_poblacin_beneficiada_y_cantidad_esperada)
(assigned_to t_definir_poblacin_beneficiada_y_cantidad_esperada proyecto_principal)
(pending t_definir_novedad_de_la_propuesta)
(assigned_to t_definir_novedad_de_la_propuesta proyecto_principal)
(pending t_definir_impacto_esperado)
(assigned_to t_definir_impacto_esperado proyecto_principal)
(pending t_definir_productos_esperados)
(assigned_to t_definir_productos_esperados proyecto_principal)
(pending t_revisar_coherencia_y_completar_la_propuesta)
(assigned_to t_revisar_coherencia_y_completar_la_propuesta proyecto_principal)
(pending t_ajustar_contenido_y_anexos_de_la_propuesta)
(assigned_to t_ajustar_contenido_y_anexos_de_la_propuesta proyecto_principal)
(pending t_enviar_propuesta_a_la_convocatoria_interna)
(assigned_to t_enviar_propuesta_a_la_convocatoria_interna proyecto_principal)
)
(:htn :tasks (t1 (gestionar_proyecto_principal proyecto_principal)))
(:goal (and (completed t_recopilar_informacin_del_proyecto) (completed t_diligenciar_datos_generales_ttulo_modalidad_coordinador_duracin_fechas_lugar) (completed t_redactar_descripcin_problema_y_antecedentes) (completed t_definir_articulacin_con_planes_institucionales_pdi_ppu_plan_facultaddepartamento) (completed t_definir_objetivo_general_y_objetivos_especficos) (completed t_definir_metodologa_diseo_desarrollo_evaluacin_y_oferta) (completed t_definir_catlogoportafolio_de_educacin_continuada_cursos_diplomados_talleres) (completed t_definir_poblacin_beneficiada_y_cantidad_esperada) (completed t_definir_novedad_de_la_propuesta) (completed t_definir_impacto_esperado) (completed t_definir_productos_esperados) (completed t_revisar_coherencia_y_completar_la_propuesta) (completed t_ajustar_contenido_y_anexos_de_la_propuesta) (completed t_enviar_propuesta_a_la_convocatoria_interna)))
)

5. Results

 doi:10.20944/preprints202602.1153.v1

The research demonstrates a platform that combines an HTN planning environment with the capture of base knowledge for any university project represented in the BPMN standard. This platform produces plans that help project managers choose the best approach for implementing a project at a university. Project formulators or those responsible for executing university projects can obtain implementation plans for different university contexts through the system, which is user-centered. Rather than requiring a planning expert to model actions and problems, the platform accepts a BPMN diagram in .xml format. By mapping the diagram, the platform captures approximately 98 % of the data and automatically generates multiple plans for a project. These plans incorporate activity costs and complexity parameters provided by the project formulators, allowing the best plan to be selected for implementation. The system automates the generation of diverse project implementation plans, enabling university project planners to focus on decision-making and critical analysis. This is achieved by incorporating cost metrics and visualizing plan length and execution time using an HTN planners SHOP 2 and PyHOP in this research.

The research demonstrates how the PlanProjU system integrates an HTN planning environment with a BPMN knowledge source specifically for university projects. It generates a variety of plans based on different metrics, thereby supporting project implementation within the university context. Project developers can automate implementation or execution by generating plans, while the system remains focused on the user's needs. Usability tests were carried out with 3 public universities in Colombia. For the UNAL-Audifarma project, the following BPMN was generated on the bpmn.io platform, as shown in Figure 5.

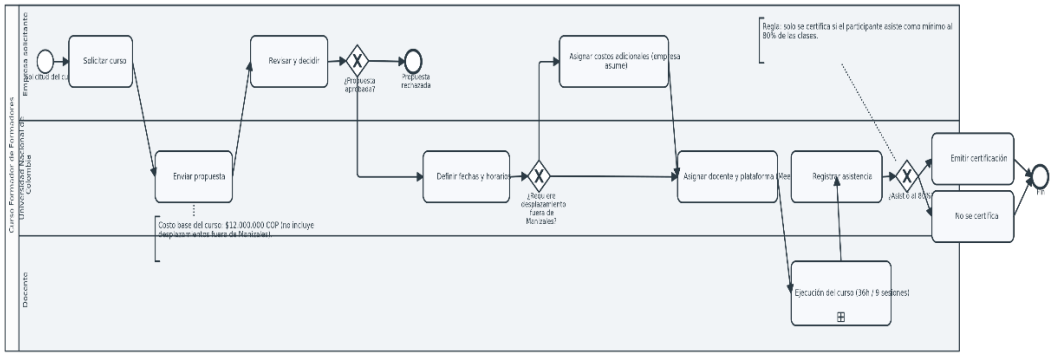

Figure 5. Project UNAL-Audifarma. Source: self-made.

Figure 5. Represents a BPMN model of the Audifarma-UNAL Train-the-Trainer extension project, structured by pools/lanes that define responsibilities (requestor, course coordination/management, and teaching/delivery role). The flow begins with the course request and the preparation/submission of the proposal, followed by a review and decision activity supported by a dedicated gateway, which branches the process toward approval or closure due to rejection. In case of approval, the model incorporates planning activities such as defining dates and resource allocation, as well as a gateway that specifies logistical and/or cost conditions, articulated with the rule that additional costs are assumed by the company. Then, the delivery of the course (3 hours per session and 5 sessions) is connected to the attendance record, which acts as compliance control: a final gateway applies the certification policy, where a certificate is only issued if the participant attends at least 80% of the classes; Otherwise, it leads to the event of non-certification. Overall, the diagram demonstrates a process with explicit decision points, cost traceability, and a verification mechanism (attendance) that ensures consistency between the provision of the training service and the institutional certification criteria.

The interaction with each official took approximately 1 to 1.5 hours, given the interruptions during the platform demonstration meeting. Each official was given a tour to navigate the platform and informed that a question mark in the upper right corner contained a guide developed within the platform to help them build the BPMN. During the interactions, some officials felt it necessary to

 doi:10.20944/preprints202602.1153.v1

view the file in BPMN or XML format. They were informed that this visualization was not available on the PlanProjU platform, as it focuses on HTN-type artificial intelligence planning and not on generating diagram visualizations under the BPMN standard. However, it is made clear to them that with the bpmn.io tool they can load the generated bpmn or xml file to visualize and modify the diagram, and that the integration of a visualization module into the platform's architecture is considered as future research work. Figure 6 shows the plans generated in PlanProjU.

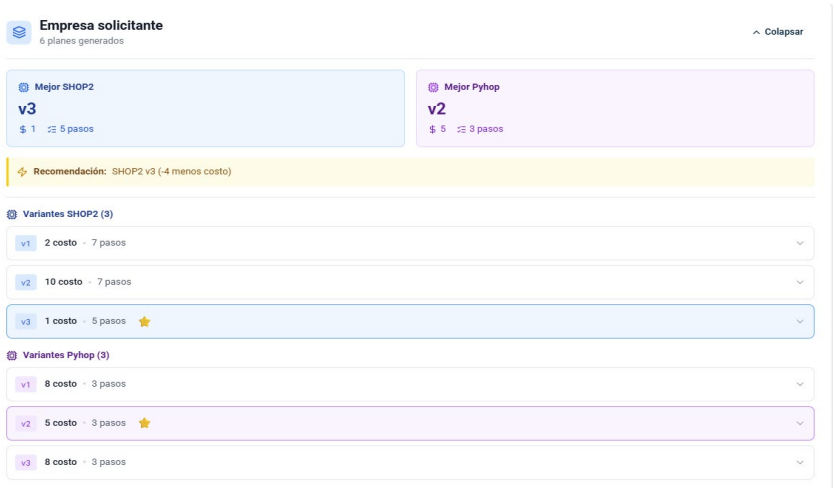

Figure 6. Generates Plan for UNAL-Audifarma. Source: self-made.

Figure 6. The output of PlanProjU for the Audifarma - UNAL Trainer Training case study is shown. The platform generates six plans, with three variants for each planner: SHOP2 and PyHOP. It presents, in summary form, the best plan from each algorithm along with an overall recommendation based on the lowest cost criterion. This organization is relevant for validation because it transforms the result of the BPMN 2.0 process into a set of comparable alternatives based on cost and step length metrics. This allows the end user or public official to evaluate not only a single plan but also the stability of the solution against different search/heuristic variants.

Analyzing the National University of Colombia block reveals a clear trade-off between economic efficiency and plan length: SHOP2 is selected as the best in its family with a cost of 6 and 7 steps, while Pyhop v3 has a cost of 7 with 4 steps. Consequently, the platform recommends SHOP2 because it is one unit cheaper, even though the plan is longer. This behavior suggests that, for this project, the search process penalizes certain decisions more significantly than simply reducing the number of activities. In terms of interpretation, Pyhop seems to produce plans with fewer steps but a higher overall cost; SHOP2, on the other hand, achieves preferable solutions, which is consistent with scenarios were breaking down activities further allows for choosing less expensive alternatives.

Finally, when comparing the three panels Requesting Company, National University, and Professor it is clear that the platform maintains the same decision logic and explicitly highlights the sensitivity of the result: for example, in the Requesting Company panel, the recommendation favors SHOP 2 due to a marked cost difference of 1 vs. 5, while in the Professor panel, the advantage of SHOP 2 also remains (5 vs. 7), although with shorter plans of 4–5 steps. This cross-cutting consistency reinforces the usefulness of PlanProjU as a decision support tool, since the user, as an extension office staff member, can justify the selection of the recommended plan based on quantifiable differences. Furthermore, they can examine alternatives (v1–v3) when cost is not the only relevant criterion. Taken together, the figures demonstrate that the validation process is not limited to generating plans but also enables a reproducible comparative analysis between the plans generated by the planners.

Table 1. Plan found for project to UNAL-Audifarma with PyHOP.

Plan 1	Plan 2	Plan 3

 doi:10.20944/preprints202602.1153.v1

client_need_management3 proposal_presentation3 technical_and_financial_analysis_of_the_proposal3 request_for_approval_from_the_research_and_extension_committee6 approval_by_the_faculty_council6 campus_council_approval_if_applicable_	client_need_management3 proposal_presentation3 approval_by_the_faculty_council3 campus_council_approval_if_applicable_	client_need_management3 proposal_presentation3 verify_proposal_presentation2 technical_and_financial_analysis_of_the_proposal6 request_for_approval_from_the_research_and_extension_committee6 verify_request_for_approval_from_the_research_and_extension_committee4 approval_by_the_faculty_council9 campus_council_approval_if_applicable_

Table 2. Plan found for project to UNAL-Audifarma with SHOP 2.

Plan 1	Plan 2	Plan 3
execute_request_for_approval_from_the_research_and_extension_committee34	execute_technical_and_financial_analysis_of_the_proposal38	execute_proposal_presentation33
execute_technical_and_financial_analysis_of_the_proposal40	execute_approval_by_the_faculty_council46	execute_campus_council_approval_if_applicable29
execute_client_need_management40	execute_technical_and_financial_analysis_of_the_proposal38	execute_campus_council_approval_if_applicable45
execute_campus_council_approval_if_applicable	execute_approval_by_the_faculty_council45 execute_proposal_presentation39 execute_request_for_approval_from_the_research_and_extension_committee	execute_client_need_management

Figure 7 shows the metrics for the plans generated for the UNAL-AUDIFARMA extension project.

Figure 7 provides a quantitative side-by-side comparison of the plans generated by the two HTN planners (SHOP 2 vs. PyHOP) using two decision metrics: total cost and number of steps. In the upper chart, SHOP 2 achieves a substantially lower aggregate cost than PyHOP, and the dashboard explicitly reports a cost gap of 33 units, indicating that under the same problem encoding and cost model SHOP 2 produces a solution that is more cost-efficient for this case. In the lower chart SHOP 2 also returns a shorter plan (fewer actions) than PyHOP, which suggests a more compact decomposition and/or fewer required operator applications to reach the goal state. Taking together, these results support the platform's recommendation logic: SHOP 2 is preferable here because it dominates PyHOP on both criteria (lower cost and fewer steps), which typically implies reduced execution effort, lower resource consumption, and a simpler implementation trajectory for the underlying BPMN-derived project activities. For the Universidad de Caldas project, the following BPMN was generated on the bpmn.io platform, as shown in Figure 8.

 doi:10.20944/preprints202602.1153.v1

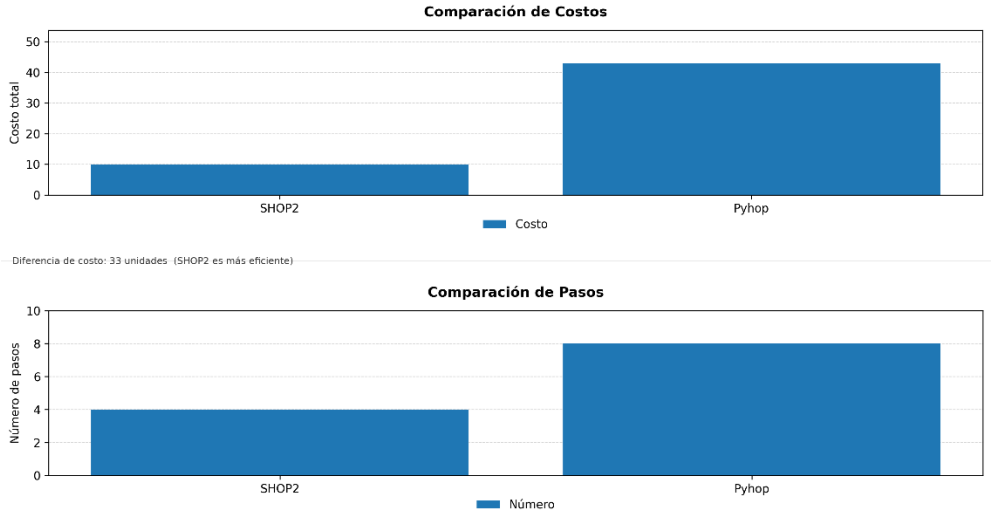

Figure 7. Metrics Plan for UNAL-Audifarma project. Source: self-made.

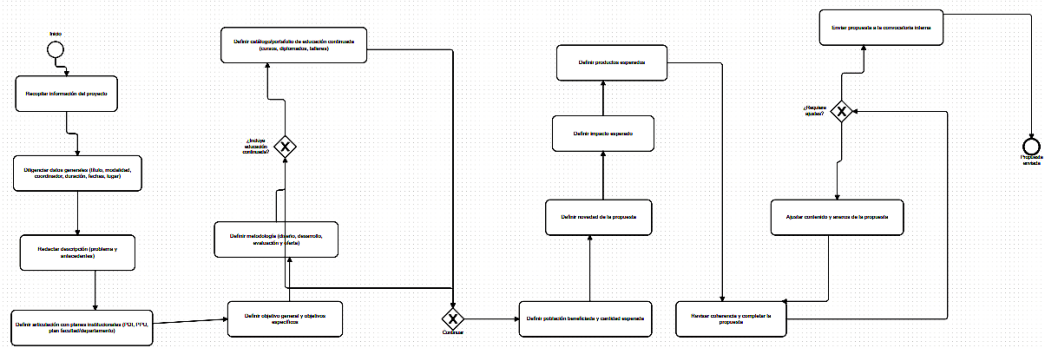

Figure 8. BPMN for Universidad de Caldas project. Source: self-made.

Figure 8. Represents an end-to-end BPMN flow that structures the process as a chain of sequential activities, interrupted by decision points that determine alternative paths and control the case's evolution until its closure. In modeling terms, the diagram shows a defined main path (start → execution of operational tasks → completion), complemented by conditional branches that introduce exception or validation scenarios. This architecture is relevant to the thesis because it allows us to justify that the model is not limited to listing tasks, but rather captures business logic, dependencies, and continuity criteria, which increases the process's fidelity to the actual operation. Furthermore, the presence of explicit decisions facilitates the transition to automated planning: each task can be formalized as an action (with preconditions and effects), while the decision points translate into conditions that define multiple execution paths and enable the comparison of solutions in terms of cost and number of steps, aligning with the objective of evaluating implementation variants within the platform. Figure 9. Shows the plans generated in PlanProjU.

 doi:10.20944/preprints202602.1153.v1

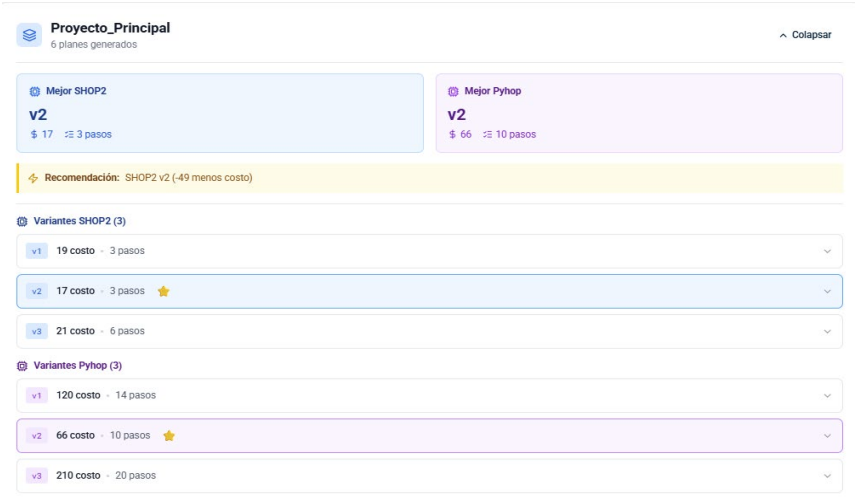

Figure 9. Generates plan for Universidad de Caldas project. Source: self-made.

Figure 9. Shows the platform's plan comparison module, where the best plan generated by SHOP 2 is compared to the best plan generated by PyHOP in terms of total cost (objective function) and number of steps. In this case, SHOP 2 selects the option with a cost of 19 and 3 steps as optimal, while PyHOP reports its variant with a cost of 66 and 10 steps as best. This supports the interface's automatic recommendation in favor of SHOP 2, as it is 49 units less expensive. The breakdown of the SHOP 2 plan shows a concrete and traceable sequence of activities, suggesting that the plan prioritizes a minimal procedure to reach the target state with the lowest aggregate cost. Similarly, the presence of multiple variants per planner reveals a range of solutions with cost-length trade-offs: SHOP 2 offers alternatives with simultaneous variations in cost and steps, while Pyhop maintains a constant length of 10 steps but with different costs (17 and 66), consistent with differences in search strategy and task decomposition. Overall, the figure supports the idea that the platform generates plans and allows users to select the best plan based on cost and step length metrics.

Table 3. Plan found for project to Universidad de Caldas with PyHOP.

Plan 1	Plan 2	Plan 3
collect_project_information3	collect_project_information3	collect_project_information3
fill_in_general_data__title__	define_alignment_with_institf	ill_in_general_data__title__
modality__coordinator__dur	utional_plans__PDI__PPU__fm	odality__coordinator__dur
ation__dates__location_3	aculty_department_plan_3	ation__dates__location_3
		verify_fill_in_general_data__
write_description__problem_	define_general_objective_an	title__modality__coordinator
and_background_3	d_specific_objectives3	__duration__dates__location
		_2
define_alignment_with_institf	define_continuing_education	write_description__problem_
utional_plans__PDI__PPU__f	_portfolio_catalog__courses_	and_background_6
aculty_department_plan_6	_diplomas__workshops_6	
define_general_objective_an	define_beneficiary_populatio	define_alignment_with_instit
d_specific_objectives6	n_and_expected_quantity6	utional_plans__PDI__PPU__f
		aculty_department_plan_6
define_methodology__desig	define_novelty_of_the_propo	verify_define_alignment_wit
n__development__evaluatio	sal6	h_institutional_plans__PDI__
n_and_offering_6		PPU__faculty_department_p
		lan_4

 doi:10.20944/preprints202602.1153.v1

define_continuing_education_portfolio_catalog__courses__diplomas__workshops_9	define_expected_impact9	define_general_objective_and_specific_objectives9
define_beneficiary_population_and_expected_quantity9	define_expected_products9	define_methodology__design__development__evaluation_and_offering_9
define_novelty_of_the_proposal9	adjust_content_and_attachments_of_the_proposal9	verify_define_methodology__design__development__evaluation_and_offering_6
define_expected_impact12	submit_proposal_to_the_internal_call	define_continuing_education_portfolio_catalog__courses__diplomas__workshops_12
review_coherence_and_complete_the_proposal12	define_novelty_of_the_proposal6	define_beneficiary_population_and_expected_quantity12
adjust_content_and_attachments_of_the_proposal15		verify_define_beneficiary_population_and_expected_quantity8
submit_proposal_to_the_internal_call		define_novelty_of_the_proposal15
		define_expected_impact15
		verify_define_expected_impact10
		define_expected_products18
		review_coherence_and_complete_the_proposal18
		verify_review_coherence_and_complete_the_proposal12
		adjust_content_and_attachments_of_the_proposal21
		submit_proposal_to_the_internal_call

Table 4. Plan found for project to Universidad de Caldas with SHOP 2.

Plan 1	Plan 2	Plan 3
execute_review_coherence_and_complete_the_proposal38	execute_write_problem_description_and_background45	execute_collect_project_information42
execute_define_beneficiary_population_and_expected_quantity30	execute_collect_project_information38	execute_define_expected_impact39
execute_fill_in_general_data_title_modality_coordinator_duration_dates_location	execute_define_general_objective_and_specific_objectives35	execute_fill_in_general_data_title_modality_coordinator_duration_dates_location
		execute_adjust_content_and_attachments_of_the_proposal33
		execute_define_expected_impact29
		execute_define_general_objective_and_specific_objectives

Figure 10 Shows the metrics for the plans generated for the Universidad de Caldas extension project.

 doi:10.20944/preprints202602.1153.v1

Figure 10. Metrics for Universidad de Caldas project. Source: self-made.

Figure 10. reinforces the comparative evaluation between SHOP2 and PyHOP using the same two decision criteria: total cost and plan length (number of steps). In the upper panel the platform reports a cost difference of 189 units, with SHOP 2 producing a markedly lower-cost plan than Pyhop; this indicates that, under the configured cost function SHOP 2 finds a solution that is substantially more resource-efficient for the same BPMN-derived planning problem. In the lower panel SHOP 2 also generates a shorter plan (roughly a third of PyHOP's steps in this case), which suggests a more compact decomposition and fewer operator applications to reach the goal. Overall, the joint dominance of SHOP 2 on both metrics (lower cost and fewer steps) provides strong operational evidence for the dashboard's recommendation, since it implies lower effort, reduced cumulative resource consumption, and a simpler implementation trajectory for the project activities represented in the underlying process model.

6. Discussion

The research results introduce an innovative method that surpasses existing approaches analyzed in academic literature and offers a practical alternative for implementing university projects by converting them into BPMN process standards. These processes can then be modeled by PlanProjU to generate implementation plans for each project, supported by HTN planning with two planners: SHOP 2 and PyHOP. This method significantly improves the project implementation process in university environments by providing project developers with a robust, automated tool for generating user-centered plans. Automating the generation of plans for university project implementation streamlines decision-making regarding the necessary conditions for successful completion and promotes the use of the BPMN standard to capture all the knowledge inherent in any formulated project.

The considerable difficulty of formulating a project, compounded by the biases inherent in the formulation process, necessitates capturing project knowledge using a standardized process such as BPMN. Furthermore, the lack of automation in various implementation scenarios increases costs for universities when executing projects aligned with their mission and strategic plans. Therefore, an HTN planning system like PlanProjU is essential for automating project implementation, as it generates plans that provide project designers with alternative scenarios and pathways to successfully complete projects within a university environment. The evaluation shows that the system generates plans for the various problems corresponding to each process. By applying the parsing or pre-planning algorithm, a general domain is generated in HDDL, along with separate

 doi:10.20944/preprints202602.1153.v1

problems for each branch variant, gateway, and subprocess, also expressed in HDDL. The evaluation focused on confirming that the parsing algorithm preserves the domain and problem structures under the HDDL formalism for three different projects generated by two different public universities in Colombia.

The evaluated projects correspond to an extension training program for an organization known as Audifarma, developed by the National University of Colombia, Manizales campus and the creation of an innovation office, developed by the University of Caldas. Optimalness metrics such as cost and completeness (i.e., plan length) were evaluated. These metrics were visually generated by PlanProjU for each project. PlanProjU allows the average user to verify whether the domain, problem, and generated plans accurately represent the evaluated project, and to approve or reject the project. If not, the administrator can then modify the planning engines and the pre-planning algorithm.

The PlanProjU platform proved effective in automatically extracting knowledge from domain-specific projects and planning problems described in the HDDL formalism. By employing SHOP 2 and PyHOP planners, it generates multiple alternative implementations plans for each project, with plan variations based on the cost and complexity parameters of each activity. This demonstrates that PlanProjU could be adopted by any university thanks to its portability and user-centered design. Using the BPMN standard, it can model any type of project and generate diverse implementation plans in a university environment, thus eliminating formulation bias and automating implementation through HTN planning.

7. Conclusions

This study introduces several improvements over previous research. First, it captures knowledge from university projects using the BPMN standard, reducing bias in project formulation. Second, it employs an algorithm or pre-planner that translates BPMN models to automatically create the HTN planning problem and action model in the HDDL formalism. Third, the PlanProjU system focuses on usability for university project planners, including those without prior experience with HTN-type automated planning.

Fourth, it is the first work to apply HTN planning to university project management, whereas previous studies have focused on areas such as robotics, logistics, production, and disaster relief. Fifth, by using the HDDL planning standard, the system can be generically implemented in any university interested in project management. Sixth, unlike other research on HTN, this study does not rely on large language models to generate domain and application problems for comparison with planners like SHOP 2 and PyHOP. Instead, our algorithm translates an ongoing project under the BPMN standard, which in turn generates an action model and planning problem.

This allows the end user to approve or reject the generated plans based on whether they accurately represent the information of the evaluated project. The research focuses on automating the generation of plans for project execution or implementation in a university context, thus addressing the cost and bias issues that arise during formulation and implementation with the help of HTN planning. Studies like this drive innovation and scientific progress by integrating two areas of knowledge: project management and automated planning such as HTN.

The innovations highlighted in this study include several key distinctions: while existing tools leverage broad language models to refine domain construction in PDDL and HTN, they omit consideration of the HDDL formalism and limit their testing to well-known domains, such as logistics, robotics, transportation, and risk management, for plan generation. In contrast, PlanProjU incorporates HDDL formalism, analyzing a new domain such as university projects to generate multiple plans using SHOP 2 and PyHOP planners, and offers an intuitive interface designed for non-technical users, including university project developers, thus opening new avenues for research in HTN planning and project management. Experimental validation with two universities and three different generated projects demonstrated that PlanProjU reduced the time required for project implementation. This efficiency was due to the automation of 98% of the knowledge gathered for projects conceived by extension offices. The resulting plans were endorsed and approved by

 doi:10.20944/preprints202602.1153.v1

extension officers at the Colombian universities in the case studies. These quantitative findings demonstrate that PlanProjU not only automates the execution and implementation of university projects but also enables the deployment of HTN artificial intelligence planning in previously unexplored contexts, generating effective results in project planning.

8. Future work

Future research will aim to consolidate and broaden the methodological framework that merges Business Process Modeling and Notation (BPMN) with Hierarchical Task Network (HTN) planning to automate university outreach projects; although the present system successfully converts BPMN process structures into HDDL domains, additional effort is required to refine the semantic representation and contextual nuances of actors, resources, and control-flow dependencies, and by incorporating domain ontologies and knowledge graphs we can achieve a more accurate alignment between BPMN entities and HTN constructs, thereby enhancing the expressiveness of the resultant planning models.

Finally, the research agenda calls for integrating neuromyotonic learning methods to boost the system's adaptability and predictive power; by blending machine-learning techniques with HTN symbolic reasoning the system can automatically learn action models, cost estimates, and decomposition patterns directly from project execution logs, and by leveraging large-language models and transformer-based architectures it can extract procedural knowledge from textual project documents.

Author Contributions: For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualization, JWSO. and NDDM.; methodology, JWSO.; software, NDDM.; validation, JWSO, NDDM. and LFCO.; formal analysis, JWSO.; investigation, LFCO.; resources, NDDM.; data curation, JWSO.; writing—original draft preparation, JWSO.; writing—review and editing, NDDM.; visualization, JWSO.; supervision, LFCO.; project administration, LFCO.; funding acquisition, LFCO. All authors have read and agreed to the published version of the manuscript." Please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Funding: Please add: This research was funded by the Allocation for Science, Technology, and Innovation of the Colombian General Royalties System through the project "Formation of High-Level Human Capital, University of Caldas" (code BPIN 2019000100035). and "The APC was funded by Universidad de Caldas". Check carefully that the details given are accurate and use the standard spelling of funding agency names at https://search.crossref.org/funding. Any errors may affect your future funding.

Data Availability Statement: We encourage all authors of articles published in MDPI journals to share their research data. In this section, please provide details regarding where data supporting reported results can be found, including links to publicly archived datasets analyzed or generated during the study. Where no new data was created, or where data is unavailable due to privacy or ethical restrictions, a statement is still required. Suggested Data Availability Statements are available in section "MDPI Research Data Policies" at https://www.mdpi.com/ethics.

Acknowledgments: The authors gratefully acknowledge the Valencian Research Institute for Artificial Intelligence (VRAIN), Universitat Politècnica de València, for providing facilities, guidance, and learning opportunities that were instrumental to the development of this research.

Conflicts of Interest: We hereby affirm that we have no significant competing interests, whether financial or non-financial, professional, or personal, that could influence the impartial and comprehensive presentation of the work detailed in this manuscript.

 doi:10.20944/preprints202602.1153.v1

Abbreviations

The following abbreviations are used in this manuscript:

AP	Automated Planning
HTN	Hierarchical Task Network
BPMN	Business Project Management Notation
SHOP 2	Simple Hierarchical Ordered Planner 2
PyHOP	Python Planner
XML	Extensible Markup Language
JSON	JavaScript Object Notation
LLM	Large Language Model
AI	Artificial Intelligence
DSR	Design Science Research
PMP	Project Management Professional
HDDL	Hierarchical Domain Definition Language
PDDL	Planning Domain Definition Language
BP	Business Process
PMS	Process Management Scheduling
DSR	Design Science Research
POTD	Planning Operator Transfer and Decomposition

References

1. D. S. Nau *et al.*, "SHOP2: An HTN planning system," *J. Artif. Intell. Res.*, vol. 20, pp. 379–404, 2003.
2. M. Ghallab, D. Nau, and P. Traverso, "Automated planning," *Theory Pract.*, 2014.
3. M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: theory and practice*. Elsevier, 2004.
4. R. Pellerin and N. Perrier, "A review of methods, techniques and tools for project planning and control," *Int. J. Prod. Res.*, vol. 57, no. 7, pp. 2160–2178, Apr. 2019, doi: 10.1080/00207543.2018.1524168.
5. Ö. Hazır, "A review of analytical models, approaches and decision support tools in project monitoring and control," *Int. J. Proj. Manag.*, vol. 33, no. 4, pp. 808–815, 2015.
6. W. Herroelen, "Project scheduling—Theory and practice," *Prod. Oper. Manag.*, vol. 14, no. 4, pp. 413–432, 2005.
7. W. Herroelen and R. Leus, "Robust and reactive project scheduling: a review and classification of procedures," *Int. J. Prod. Res.*, vol. 42, no. 8, pp. 1599–1620, 2004.
8. J. Węglarz, J. Józefowska, M. Mika, and G. Waligóra, "Project scheduling with finite or infinite number of activity processing modes–A survey," *Eur. J. Oper. Res.*, vol. 208, no. 3, pp. 177–205, 2011.
9. J. Zhou, P. E. D. Love, X. Wang, K. L. Teo, and Z. Irani, "A review of methods and algorithms for optimizing construction scheduling," *J. Oper. Res. Soc.*, vol. 64, no. 8, pp. 1091–1105, 2013.
10. Y. Zhen *et al.*, "LLM-Project: Automated Engineering Task Planning via Generative AI and WBS Integration," in *2024 IEEE 14th International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2024, pp. 605–610. doi: 10.1109/CYBER63482.2024.10749328.
11. M. Schroder, "Autoscrum: Automating project planning using large language models," *arXiv Prepr. arXiv2306.03197*, 2023.
12. D. Höller, P. Bercher, G. Behnke, and S. Biundo, "HTN planning as heuristic progression search," *J. Artif. Intell. Res.*, vol. 67, pp. 835–880, 2020.
13. K. Erol, J. Hendler, and D. S. Nau, "Complexity results for HTN planning," *Ann. Math. Artif. Intell.*, vol. 18, no. 1, pp. 69–93, 1996.
14. P. Bercher, R. Alford, and D. Höller, "A Survey on Hierarchical Planning-One Abstract Idea, Many Concrete Realizations.," in *IJCAI*, 2019, pp. 6267–6275.
15. D. Höller *et al.*, "HDDL: An extension to PDDL for expressing hierarchical planning problems," in *Proceedings of the AAAI conference on artificial intelligence*, 2020, pp. 9883–9891.

16. A. Taitler *et al.*, "The 2023 international planning competition," 2024, *Wiley Online Library*.
17. F. W. Taylor, *Scientific management*. Routledge, 2004.
18. M. Klun and P. Trkman, "Business process management–at the crossroads," *Bus. Process Manag. J.*, vol. 24, no. 3, pp. 786–813, 2018.
19. B. Heinrich, F. Krause, and A. Schiller, "Automated planning of process models: The construction of parallel splits and synchronizations," *Decis. Support Syst.*, vol. 125, 2019, doi: 10.1016/j.dss.2019.113096 WE - Science Citation Index Expanded (SCI-EXPANDED).
20. A. Marrella, M. Mecella, and S. Sardina, "Intelligent process adaptation in the SmartPM system," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 2, 2017, doi: 10.1145/2948071.
21. D. Nau, H. Munoz-Avila, Y. Cao, A. Lotem, and S. Mitchell, "Total-order planning with partially ordered subtasks," in *IJCAI*, 2001, pp. 425–430.
22. D. Nau, Y. Cao, A. Lotem, and H. Munoz-Avila, "SHOP: Simple hierarchical ordered planner," in *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, 1999, pp. 968–973.
23. D. Nau, "Game applications of HTN planning with state variables," in *Planning in Games: Papers from the ICAPS Workshop*, 2013.
24. H. Schuschel and M. Weske, "Triggering replanning in an integrated workflow planning and enactment system," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3255, pp. 322–335, 2004, doi: 10.1007/978-3-540-30204-9_22.
25. M. D. R-Moreno, D. Borrajo, A. Cesta, and A. Oddi, "Integrating planning and scheduling in workflow domains," *Expert Syst. Appl.*, vol. 33, no. 2, pp. 389–406, 2007, doi: 10.1016/j.eswa.2006.05.027.
26. R. F. Da Silva, R. Filgueira, I. Pietri, M. Jiang, R. Sakellariou, and E. Deelman, "A characterization of workflow management systems for extreme-scale applications," *Futur. Gener. Comput. Syst.*, vol. 75, pp. 228–238, 2017.
27. H. A. Reijers, I. Vanderfeesten, and W. M. P. van der Aalst, "The effectiveness of workflow management systems: A longitudinal study," *Int. J. Inf. Manage.*, vol. 36, no. 1, pp. 126–141, 2016.
28. A. Marrella, "Automated planning for business process management," *J. Data Semant.*, vol. 8, no. 2, pp. 79–98, 2019.
29. M. La Rosa, W. M. P. Van Der Aalst, M. Dumas, and F. P. Milani, "Business process variability modeling: A survey," *ACM Comput. Surv.*, vol. 50, no. 1, pp. 1–45, 2017.
30. D. V McDermott, "Estimated-Regression Planning for Interactions with Web Services.," in *AIPS*, 2002, pp. 204–211.
31. M. Sheshagiri and T. Finin, "A planner for composing services described in DAML-S," in *Proceedings of the AAMAS Workshop on Web Services and Agent-based Engineering*, 2003.
32. J. Peer, "A pop-based replanning agent for automatic web service composition," in *European semantic web conference*, Springer, 2005, pp. 47–61.
33. M. Klusch and A. Gerber, "Fast composition planning of owl-s services and application," in *2006 European Conference on Web Services (ECOWS'06)*, IEEE, 2006, pp. 181–190.
34. R. P. A. Petrick and F. Bacchus, "Extending the Knowledge-Based Approach to Planning with Incomplete Information and Sensing.," in *ICAPS*, 2004, pp. 2–11.
35. J. Hoffmann, I. Weber, and F. Kraft, "Sap speaks PDDL," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010, pp. 1096–1101.
36. S. McIlraith and T. C. Son, "Adapting golog for composition of semantic web services," *Kr*, vol. 2, no. 200, p. 2, 2002.
37. S. Sohrabi, N. Prokoshyna, and S. A. McIlraith, "Web service composition via generic procedures and customizing user preferences," in *International Semantic Web Conference*, Springer, 2006, pp. 597–611.
38. P. Traverso and M. Pistore, "Automated composition of semantic web services into executable processes," in *International Semantic Web Conference*, Springer, 2004, pp. 380–394.
39. P. Bertoli, M. Pistore, and P. Traverso, "Automated Web Service Composition by On-the-Fly Belief Space Search.," in *ICAPS*, 2006, pp. 358–361.
40. M. Yousefi and P. Bercher, "HDDL Parser: A Realtime Hierarchical Planning Language Validation Toolkit," 2025.

 doi:10.20944/preprints202602.1153.v1

41. S. Stein, S. Kühne, and K. Ivanov, "Business to it transformations revisited," in *International Conference on Business Process Management*, Springer, 2008, pp. 176–187.
42. J. Vanhatalo, H. Völzer, and J. Koehler, "The refined process structure tree," *Data Knowl. Eng.*, vol. 68, no. 9, pp. 793–818, 2009.
43. R. Barták, J. Little, O. Manzano, and C. Sheahan, "From enterprise models to scheduling models: bridging the gap," *J. Intell. Manuf.*, vol. 21, no. 1, pp. 121–132, 2010.
44. U. Köckemann, "AIDDL: The AI Domain Definition Language for integrated AI systems," *SoftwareX*, vol. 31, no. February, p. 102259, 2025, doi: 10.1016/j.softx.2025.102259.
45. R. Alford, V. Shivashankar, U. Kuter, and D. Nau, "HTN problem spaces: Structure, algorithms, termination," in *Proceedings of the International Symposium on Combinatorial Search*, 2012, pp. 2–9.
46. P. Bercher, G. Behnke, D. Höller, and S. Biundo, "An Admissible HTN Planning Heuristic.," in *IJCAI*, 2017, pp. 480–488.
47. T. Geier and P. Bercher, "On the decidability of HTN planning with task insertion," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, 2011, p. 1955.
48. D. Nau *et al.*, "Applications of SHOP and SHOP2," *IEEE Intell. Syst.*, vol. 20, no. 2, pp. 34–41, 2005.
49. A. Dresch, D. P. Lacerda, and J. A. V. Antunes, *Design science research: A method for science and technology advancement*. 2015. doi: 10.1007/978-3-319-07374-3.
50. S. Siedhoff, "Design science research," pp. 29–43, 2019, doi: 10.1007/978-3-658-26336-2_3.
51. K. Owczarzak, J. Conroy, H. T. Dang, and A. Nenkova, "An assessment of the accuracy of automatic evaluation in summarization," in *Proceedings of workshop on evaluation metrics and system comparison for automatic summarization*, 2012, pp. 1–9.