From Formalisers to Engineers: An Agentic Position to HTN Domain Model Generation

Anonymous Submission

Abstract

The ability of HTN planning to exploit rich domain knowledge is both its greatest strength and its primary limitation. While such knowledge can improve planning efficiency, encoding it explicitly in formal domain models is complex, time-consuming, and demands expertise in the application domain and automated planning. Recent LLM-based approaches largely frame this challenge as translating natural-language descriptions into PDDL formal models. For HTN planning, however, the problem is deeper: task hierarchies depend on procedural knowledge that is often tacit and difficult to obtain reliably from text alone. We therefore argue that HTN domain modelling should be treated not as a formalisation problem, but as a knowledge-engineering process. To this end, we propose an architecture in which LLM-based agents assume responsibility for the typical stages of knowledge engineering for planning to collaboratively construct HTN domain models from natural-language input. We conclude with open research questions and a three-stage roadmap.

1 The Problem

AI planning is a subfield of artificial intelligence uniquely positioned by its focus on providing goal-oriented behaviour. This is realised by searching, selecting, and organising actions required to achieve a user's objective when executed in a given initial state of a targeted environment. The field focuses on the design and development of planning systems that require a symbolic model of the world, the user's objective, and the initial state. AI planning is thus a model-based approach that critically depends on the systematic acquisition, formalisation, and representation of the multifaceted knowledge about the targeted application domain in a *domain model*, enabling planning systems to reason over this knowledge and generate plans executable in operational settings.

Depending on the planning technique, the domain model can have different structures and require different kinds of knowledge. Classical planning relies on action schemas with preconditions and effects, whereas Hierarchical Task Network (HTN) planning goes further and employs hierarchical structures that encode procedural domain knowledge [Georgievski and Aiello, 2015]. It uses primitive tasks (actions as in classical planning), and compound tasks, which require decomposition into sub-tasks. Domain knowledge is encoded as methods that describe how to decompose compound tasks into task networks, i.e., structured representations of tasks and their ordering. This ability to represent recipe-like decompositions has enabled HTN planning to succeed in real-world domains such as autonomous vehicles [Alnazer *et al.*, 2022], building automation [Georgievski *et al.*, 2017], and playing poker [Alnazer *et al.*, 2025b].

Engineering domain models is a difficult and labour-intensive activity, constituting a major barrier to the adoption of AI planning systems [McCluskey *et al.*, 2017; Georgievski, 2025]. The difficulty has been recognised since the early days of AI planning, without clarity whether it is independent of whether any methodology is followed. Engineering domain models requires both domain expertise to acquire relevant knowledge and planning expertise to translate it into an adequate domain model in a planning language compatible with the selected planning tools. The combination is rarely available within a single team, and individual engineers compensate by combining personal expertise, ad hoc judgment, and the affordances of whatever tools they happen to use. The result is limited coverage of application domains, domain models that capture only a fraction of the relevant aspects of a target domain, and outcomes that are difficult to reproduce, reuse, or quality-assure [Alnazer *et al.*, 2025a].

Developing HTN domain models inherits all of these factors and exacerbates them: the additional dimension of procedural knowledge complicates the conceptualisation from domain knowledge to HTN planning constructs, requiring engineers to commit not only to *what* the world contains but to *how* tasks decompose, at multiple levels of abstraction, with constraints propagated coherently across them.

Addressing these consequences and, more broadly, reducing the engineering burden requires automating the construction of domain models. Earlier approaches, including HTN-specific work (e.g., [Fine-Morris *et al.*, 2020]) , focused on learning domain models from structured artefacts, such as execution traces, partial models, or annotated demonstrations. These methods do not reduce the engineering burden in practice: the structured artifacts they require are them-

selves products of substantial prior knowledge engineering. Also, they do not accept the form of input domain experts already use. The advance of capable Large Language Models (LLMs) reframed this problem by making natural-language input tractable as a starting point for domain model construction. A large body of work has already contributed across domain model generation, domain model editing, and benchmarking, as surveyed in [Tantakoun *et al.*, 2025]. The earliest approaches employ single-shot LLM architectures, producing domain models that lack syntactic correctness, semantic consistency, and reusability [Silver *et al.*, 2022; Xie *et al.*, 2023; Saba, 2023; Puerta-Merino *et al.*, 2025]. The dominant response to this brittleness has been neurosymbolic approaches that employ a *Generate-Verify-Correct* (GVC) pattern, where LLM domain model generation is combined with symbolic validation in feedback-driven loops [Liu *et al.*, 2023; Guan *et al.*, 2023; Oswald *et al.*, 2024; Goebel and Zips, 2025; Smirnov *et al.*, 2024; Huang and Zhang, 2025; Huang *et al.*, 2025; Muñoz-Avila *et al.*, 2025]. Two recent works attempt to move beyond this pattern. Casciani et al. [2025] incorporate knowledge elicitation via requirements engineering mechanisms into prompt construction. La Malfa et al. [2025] propose an end-to-end agentic framework but decompose agents along planning boundaries, separating grounding, plan generation, and execution monitoring rather than along knowledge-engineering concerns.

The scarcity of work on HTN domain model generation, the absence of approaches treating domain model generation as a structured engineering activity, and the lack of an agentic architecture organised around knowledge-engineering concerns define the boundaries of the research gap we address.

2 The Position

Constructing domain models for planning from natural-language descriptions is not a single step. It is a staged process: knowledge must be elicited before it can be formalised, and it must be formalised before it can be validated. This structure has long been recognised within Knowledge Engineering for Planning and Scheduling (KEPS), where the activity for domain model construction has been decomposed into acquisition, formulation, verification and validation, and maintenance [McCluskey *et al.*, 2017; Vallati *et al.*, 2025].

These stages differ in epistemic character, produce distinct typed artifacts, and fail in different ways. *Acquisition* is exploratory and produces elicited knowledge; *Formulation* is constructive and produces a structured domain model; and *Verification and Validation* is critical and produces verdicts and traces. HTN planning sharpens these stages further. The hierarchical structure of HTN domain models indicates that collapsing Formulation into a single stage is inadequate: constructing the hierarchy, which requires committing to how tasks decompose at multiple levels of abstraction, is distinct from encoding that hierarchy in the syntax of a target formalism. These are design-level and encoding-level decisions, and collapsing them loses the ability to conceptualise key HTN concepts and distinguish artefacts.

The knowledge-engineering activity also has stage-specific failure modes. A domain model may be wrong because relevant knowledge was never elicited, because elicited knowledge was poorly structured, or because the formal domain model does not match the original description. HTN planning extends the range of failures to include hierarchy-level issues. For example, *decomposition reachability*, which is the property that every compound task can eventually be decomposed into primitive tasks, may be violated when methods lead to dead ends because of the absence of applicable decomposition or decomposition chains that cycle without terminating. This makes the HTN domain model operationally useless regardless of syntactic correctness, and requires different diagnosis and repair than syntactic error does.

A GVC pipeline collapses these distinctions. It has no typed intermediate artefacts, only a context window, and no addressable repair targets. For example, when Verification and Validation detects a structural issue, the fault should be routed to the Formulation stage; when it detects a semantic mismatch, it should reach the Acquisition stage. A GVC pipeline that retries from the beginning cannot make this distinction well even with a dedicated prompt.

These observations make a multi-agent architecture a natural fit: each agent operates in its epistemic mode, writes a typed artefact that another agent reads, and can be addressed individually when repair is needed. Symbolic tools are treated as peer reasoning tools invoked during domain model construction rather than only after it. For decomposition reachability, for example, this means invoking an HTN planner during construction to return decomposition traces that localise faults, not just plan/no plan verdict at the end.

We argue that the generation of HTN planning domain models from natural-language descriptions should be approached as a structured knowledge-engineering activity and realised as an autonomous multi-agent system in which LLMs and symbolic reasoning tools cooperate as peers.

3 Architecture Overview

As a first step towards realising this position, we introduce an agentic architecture for HTN domain model generation. Figure 1 shows the architecture consisting of four agents: Acquirer, Hierarchiser, Formaliser, and Validator. Each agent takes responsibility for the stages of the knowledge-engineering process described earlier. The agents communicate through a shared workspace of typed HTN artifacts. The architecture integrates a retrieval layer over a corpus of curated sources, and a layer of symbolic tools. Four configuration resources parametrise the architecture: a set of domain knowledge categories, an HTN structural metamodel, HTN formalism specifications, and a set of acceptance criteria.

Acquirer. The Acquirer receives a natural-language description of the application domain and elicits procedural and declarative knowledge. We position elicitation here as structured acquisition along established domain knowledge categories that HTN domain models must cover, such as tasks, objectives, constraints, behaviours, and so on [Alnazer and Georgievski, 2023]. Elicitation is complete when each category has been addressed. The output, *procedural notes*, is written to the workspace. The Acquirer can query a retreival corpus of procedural sources, such as manuals, standard op-

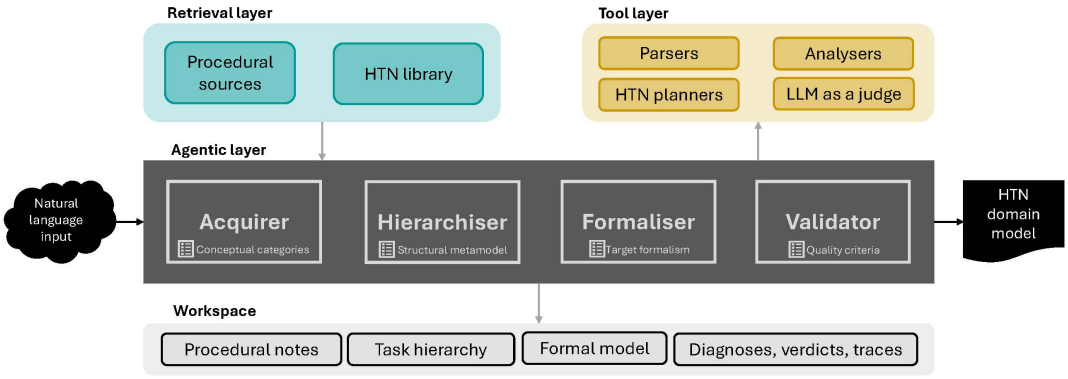

Figure 1: LLM-based agentic architecture for engineering HTN domain models.

erating procedures, and expert text, to compensate for limited LLM coverage of HTN-relevant procedural knowledge.

Hierarchiser. The Hierarchiser reads procedural notes and constructs a structured hierarchy of compound tasks, methods, primitive tasks, and their decomposition relations. It is agnostic to any HTN formalism but aware of HTN planning, as guided by an *HTN structural metamodel* that encodes the requirements of a well-formed HTN hierarchy. Retrieval over a library of validated HTN tasks can surface decomposition patterns from existing domain models.

Formaliser. The Formaliser encodes a task hierarchy in the syntax of a specific HTN formalism (e.g., HPDL, HDDL, SHOP2), consulting the target formalism specification and resolving any mismatch between hierarchy and formalism expressiveness. The separation between Hierarchiser and Formaliser allows the architecture to be re-targeted to a different HTN language by replacing only the Formaliser's configuration resource, without restructuring other agents.

Validator. The Validator reads the formal domain model, the input description, and a set of acceptance criteria that define what counts as a quality domain model for the application context. It first invokes symbolic tools to verify internal coherence (e.g., syntactic well-formedness, decomposition reachability); only if verification passes does it evaluate the domain model against the acceptance criteria, optionally using an HTN planner for executable checks and an LLM critic for others. The Validator emits a verdict and trace to the workspace; a failure is then routed to the agent best positioned to repair it, which is a diagnostic act performed by inspecting the trace, not a fixed routing table.

Autonomous loop. Each agent acts on the workspace's current state, writes its output, and the Validator's verdicts drive further agent activity until acceptance criteria are met or a budget is exhausted.

4 Open Questions and Conclusion

The architecture is agnostic about the choice of language model per agent. General-purpose LLMs are likely well-suited to the Acquirer and Formaliser, given their breadth of domain knowledge and coding capabilities; reasoning LLMs are likely better suited to the Hierarchiser and Validator, where hidden assumptions must be inferred and structural consistency enforced. Small Language Models (SLMs) [Belcak *et al.*, 2025] could offer reliability on narrow tasks (e.g., syntax generation) but would require training data.

We deliberately leave the loop's control mechanism open; what aspects are sequential or concurrent, and how termination is managed beyond convergence and budget exhaustion, are open design choices. Also, whether the Validator's routing of failures can be performed reliably by an LLM, a symbolic tool, or a hybrid is an open question.

Whether the proposed architecture, once instantiated, outperforms LLM-based pipelines, such as NL2HTN [Puerta-Merino *et al.*, 2025], can only be answered by comparing it against benchmarks designed for generation rather than planning. Evaluating LLM-based generation requires paired natural-language descriptions and reference domain models, which is currently missing for HTN planning. Beyond raw performance, operationalising knowledge-engineering quality dimensions as automated metrics, and distinguishing those that can be automated from those that require human evaluation, remains an open methodological question.

We see a three-stage path toward realising the position. The first stage is infrastructure: a curated retrieval corpus, operationalised acceptance criteria, paired natural-language benchmarks built on existing IPC HTN domains, and a curated suite of symbolic tools. The second is instantiation: a working system on a single HTN formalism, with conservative loop control and general-purpose LLMs as agents, evaluated against single-shot LLM-based pipelines. The third is refinement: reasoning LLMs, specialised SLMs, richer mechanisms for the autonomous loop, and extension across HTN formalisms.

References

[Alnazer and Georgievski, 2023] Ebaa Alnazer and Ilche Georgievski. Understanding Real-World AI Planning Domains: A Conceptual Framework. In *Symposium and Sum. Sch. on Service-Oriented Computing*, pages 3–23, 2023.

[Alnazer *et al.*, 2022] Ebaa Alnazer, Ilche Georgievski, Neha Prakash, and Marco Aiello. A Role for HTN Planning in Increasing Trust in Autonomous Driving. In *IEEE Int'l Smart Cities Conference*, pages 1–7, 2022.

[Alnazer *et al.*, 2025a] Ebaa Alnazer, Ilche Georgievski, and Marco Aiello. Risk-Aware HTN Planning Domain Models for Autonomous Vehicles and Satellites. In *Int'l Conf. on AI-based Systems and Services*, pages 36–45, 2025.

[Alnazer *et al.*, 2025b] Ebaa Alnazer, Ilche Georgievski, and Marco Aiello. Texas Hold'em Meets Risk-Aware HTN Planning: A Modelling and Architectural Perspective. In *ECAI Workshop on AI-based Planning for Complex Real-World Applications*, pages 15–24, 2025.

[Belcak *et al.*, 2025] Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small Language Models are the Future of Agentic AI. Preprint 2506.02153, arXiv, 2025.

[Casciani *et al.*, 2025] Angelo Casciani, Giuseppe De Giacomo, Andrea Marrella, and Christoph Weinhuber. A Requirements Engineering-Driven Methodology for Planning Domain Generation via LLMs with Invariant-Based Refinement. In *ICAPS Workshop on Planning in the Era of LLMs*, 2025.

[Fine-Morris *et al.*, 2020] Morgan Fine-Morris, Bryan Auslander, Michael W Floyd, Greg Pennisi, Héctor Muñoz-Avila, and EDU Kalyan Moy Gupta. Learning Hierarchical Task Networks with Landmarks and Numeric Fluents by Combining Symbolic and Numeric Regression. In *Annual Conference on Advances in Cognitive Systems*, 2020.

[Georgievski and Aiello, 2015] Ilche Georgievski and Marco Aiello. HTN planning: Overview, comparison, and beyond. *Artifical Intelligence*, 222:124–156, 2015.

[Georgievski *et al.*, 2017] Ilche Georgievski, Tuan Anh Nguyen, Faris Nizamic, Brian Setz, Alexander Lazovik, and Marco Aiello. Planning meets activity recognition: Service coordination for intelligent buildings. *Perv. and Mob. Comp.*, 38(1):110–139, 2017.

[Georgievski, 2025] Ilche Georgievski. Engineering AI Planning Systems. Habilitation Thesis, University of Stuttgart, 2025.

[Goebel and Zips, 2025] Kai Goebel and Patrik Zips. Can LLM-Reasoning Models Replace Classical Planning? A Benchmark Study. Preprint 2507.23589, arXiv, 2025.

[Guan *et al.*, 2023] Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging Pretrained Large Language Models to Construct and Utilize World Models for Model-based Task Planning. In *Advances in Neural Information Processing Systems*, pages 79081–79094, 2023.

[Huang and Zhang, 2025] Cassie Huang and Li Zhang. On the limit of language models as planning formalizers. In *Annual Meeting of the Association for Computational Linguistics*, pages 4880–4904, 2025.

[Huang *et al.*, 2025] Sukai Huang, Nir Lipovetzky, and Trevor Cohn. Planning in the Dark: LLM-Symbolic Planning Pipeline Without Experts. In *AAAI Conference on Artificial Intelligence*, pages 26542–26550, 2025.

[La Malfa *et al.*, 2025] Emanuele La Malfa, Ping Zhu, Samuele Marro, Sara Bernardini, and Michael Wooldridge. An End-to-end Planning Framework with Agentic LLMs and PDDL. Preprint 2512.09629, arXiv, 2025.

[Liu *et al.*, 2023] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. Preprint 2304.11477, arXiv, 2023.

[McCluskey *et al.*, 2017] Thomas L McCluskey, Tiago S Vaquero, and Mauro Vallati. Engineering Knowledge for Automated Planning: Towards a Notion of Quality. In *Int'l Conference on Knowledge Capture*, pages 1–8, 2017.

[Muñoz-Avila *et al.*, 2025] Héctor Muñoz-Avila, David W Aha, and Paola Rizzo. ChatHTN: Interleaving Approximate (LLM) and Symbolic HTN Planning. Preprint 2505.11814, arXiv, 2025.

[Oswald *et al.*, 2024] James Oswald, Kavitha Srinivas, Harsha Kokel, Junkyu Lee, Michael Katz, and Shirin Sohrabi. Large language models as planning domain generators. In *Int'l Conference on Automated Planning and Scheduling*, pages 423–431, 2024.

[Puerta-Merino *et al.*, 2025] Israel Puerta-Merino, Carlos Núñez-Molina, Pablo Mesejo, and Juan Fernández-Olivares. Towards a General Framework for HTN Modeling with LLMs. 2025.

[Saba, 2023] Walid S Saba. Towards Explainable and Language-Agnostic LLMs: Symbolic Reverse Engineering of Language at Scale. Preprint 2306.00017, arXiv, 2023.

[Silver *et al.*, 2022] Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDL Planning with Pretrained Large Language Models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

[Smirnov *et al.*, 2024] Pavel Smirnov, Frank Joublin, Antonello Ceravola, and Michael Gienger. Generating Consistent PDDL Domains with Large Language Models. Preprint 2404.07751, arXiv, 2024.

[Tantakoun *et al.*, 2025] Marcus Tantakoun, Christian Muise, and Xiaodan Zhu. LLMs as planning formalizers: A survey for leveraging large language models to construct automated planning models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25167–25188, 2025.

[Vallati *et al.*, 2025] Mauro Vallati, Roman Barták, Lukáš Chrpa, Thomas L McCluskey, and Ronald PA Petrick. Knowledge Engineering for Planning and Scheduling in the LLM Era. In *Int'l Conference on Automated Planning and Scheduling*, pages 391–395, 2025.

[Xie *et al.*, 2023] Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. Translating Natural Language to Planning Goals with Large-Language Models. Preprint 2302.05128, arXiv, 2023.