UDC 004.82:519.816
DOI: 10.15587/2706-5448.2026.361385

Yaroslav Teplyi,
Dmytro Dosyn

DEVELOPMENT OF AN ONTOLOGY-BASED METHOD FOR SCALABLE GENERATION OF PARTIALLY OBSERVABLE MARKOV DECISION PROCESS (POMDP) MODELS

The object of research is the automated construction of partially observable Markov decision process (POMDP) models from a semantic knowledge base, in which probabilistic parameters are formalized as an ontology or knowledge graph.

The problem addressed in this research is that the traditional approach requires manual construction of transition, observation, reward, and cost tables, which results in low data traceability. This significantly complicates model auditing and updating, and hinders reconciliation in cases of conflicting information sources or changes in domain knowledge.

A domain-independent schema has been developed that represents each numeric POMDP parameter as a provenance-aware claim, including the information source and timestamp.

A deterministic compilation method has been implemented that selects claims via SPARQL queries, resolves conflicts according to a policy, normalizes parameters, and forms the output tabular values for transition T, observation O, and reward R matrices.

A benchmark evaluation was conducted using a diagnostic task, confirming the model's high generation speed. In a series of 10 experiments, generation time did not exceed 3.6 s, even for a state space of $|S| = 3957$, and solution time by the SARSOP solver ranged from a few seconds to a set timeout of 600 s, depending on the scaling mode.

Treating parameters as graph entities allows transforming model updates into an explicit claim-editing procedure that is accessible to audit.

The method can be used when the domain knowledge is expressed as ontology with identifiable states/actions/observations, and when parameter provenance matters. It supports maintainable decision-support deployments with auditable parameter histories and handling of conflicting claims, generation of model variants for sensitivity analysis, and controlled extension of the model via ontology learning.

Keywords: *ontology, knowledge graphs, automated model generation, probabilistic planning, POMDP.*

Received: 03.03.2026
Received in revised form: 04.05.2026
Accepted: 15.05.2026
Published: 29.05.2026

How to cite
Teplyi, Y., Dosyn, D. (2026). Development of an ontology-based method for scalable generation of Partially Observable Markov Decision Process (POMDP) models. Technology Audit and Production Reserves, 3 (2 (89)), 13–20. https://doi.org/10.15587/2706-5448.2026.361385

1. Introduction

Decision support systems in healthcare, industrial diagnosis, and cybersecurity rely on two key components typically developed separately: a structured way to represent expert knowledge and a decision model that functions under uncertainty. Ontologies and knowledge graphs provide shared vocabularies, clear semantics, and adaptable structures that evolve as knowledge advances [1, 2]. Meanwhile, partially observable Markov decision processes (POMDPs) offer a framework for making sequential decisions when the actual state is not observable and data is noisy [3]. In practical applications, both are essential: the first ensures domain assumptions remain explicit, while the second enables the computation of policies for action in uncertain situations.

Ontology-based systems excel at describing the entities, relations, and provenance within a domain, but they generally do not capture the model's probabilistic semantics. Conversely, POMDP implementations typically represent parameters as numeric tables or arrays, which are efficient for solvers but difficult to maintain. These values are hard to audit, their sources are not explicit, updates are prone to errors, and conflicting evidence is managed outside the model. The challenge increases when a model needs to be regenerated frequently due to data updates, such as new studies, revised efficacies, or cost changes, since this knowledge resides in semantic artifacts. While research on semantic modeling and POMDP planning is well-developed, their integration is still inconsistent. There is currently no standardized method for storing probabilities and utilities within a knowledge base that allows for inspection and editing while maintaining a complete, normalized model.

Ontologies and knowledge graphs are commonly used to encode expert knowledge and support integration, justifying their role as long-term repositories that evolve with new data [1, 2]. However, the question of how these semantic assets can act as a single source of truth for executable sequential decision models under uncertainty is rarely addressed. A typical approach involves using logical or ontological knowledge to guide planning or enhance observations, while keeping the core POMDP parameters external. Hybrid reasoning that combines logical and probabilistic approaches for actions with sensing has been studied extensively [4]. Incorporating commonsense reasoning into POMDP planning for dialogue and robotic behaviors has been explored [5], with recent efforts focusing on planning under logical constraints [6]. Current frameworks highlight the importance of context abnormalities and observation augmentation [7, 8], and ontology-driven reward mechanisms have been investigated in related sequential-decision

ISSN-L 2664-9969; E-ISSN 2706-5448

contexts [9]. Collectively, these approaches show that semantics can influence planning, but none treat the full POMDP parameter matrices as provenance-aware, quiryable objects within the knowledge graph.

Other studies also investigate uncertainty representations by converting ontology structures into Bayesian network frameworks [10], offering probabilistic ontology languages [11], or defining distributions over possible worlds through probabilistic axiom annotations [12, 13]. These frameworks support probabilistic reasoning within semantic models but are not intended to encode the action and reward structures needed by POMDPs. Since model generation with evolving evidence demands traceability at the parameter level, provenance must be associated with individual probabilities and utilities. While general provenance vocabularies like PROV-O [14] and publication patterns for atomic assertions [15, 16] exist, they require customization for POMDP parameters. Converting semantic descriptions into executable planning artifacts is common in deterministic environments. Tools such as OWLS-XPlan adapt OWL-S service descriptions and ontologies into planning formats for composition tasks [17]. Other ontology-based planning approaches translate or adapt OWL DL knowledge for planning interfaces [18], and domain-specific converters produce PDDL domain descriptions from semantic web services for Industry 4.0 [19]. Though these methods treat compilation as an architectural feature, their core representations remain deterministic and do not fully capture probabilistic models.

Therefore, key issues still persist in the reviewed literature. Existing research typically adopts two strategies: either ontologies are employed to provide input to external probabilistic planners without fully capturing their parameterization, or they incorporate uncertainty directly into ontologies without transforming them into a model that can be solved. The challenge of maintaining a completely unified, provenance-aware knowledge base that naturally encodes POMDP parameters and can be deterministically converted into standard solver formats remains mostly unresolved.

The object of the research is the automated construction of POMDP models from semantic knowledge bases, treating probabilistic dynamics and utility functions as formal ontological entities.

The aim of this research is to develop an ontology-based method for the scalable generation of POMDP models to automate the transformation of semantic knowledge into dynamic decision-making strategies.

The following objectives were set to achieve this aim:

1. To design a domain-independent ontology schema that represents numeric parameters as reified claims with explicit metadata, such as information source, timestamp, and context.
2. To develop a deterministic compilation procedure that extracts claim individuals via SPARQL queries and resolves competing assertions according to a configurable policy and applies the normalization of extracted parameters to form output tabular values for transition (T), observation (O), and reward (R) matrices.
3. To perform a benchmark evaluation of the proposed method on an automated diagnosis task to assess generation speed and scalability across various state spaces.

2. Materials and Methods

2.1. Mathematical foundations of POMDP

A partially observable Markov decision process (POMDP) is the common mathematical framework used for sequential decision-making tasks where the agent lacks direct access to the environment's true state. It is characterized by a tuple $(S, A, \Omega, T, O, R, \gamma)$, which includes:

- S – a finite set of unobservable environment states;
- A – a finite set of available actions;
- Ω – a finite set of possible observations;
- T: $S \times A \times S \to [0, 1]$ represents the state transition function, defining the probability of transitioning to state s' given state s and action a;
- O: $S \times A \times \Omega \to [0, 1]$ – the observation function, defining the probability of receiving observation o after taking action a and arriving in state s';
- R: $S \times A \to \mathbb{R}$ – the reward function defining the immediate expected utility of taking action a in state s;
- $\gamma \in [0, 1)$ – the discount factor applied to future rewards.

Solving a POMDP involves finding an optimal policy that maps states to probability distributions over hidden states S and actions, aiming to maximize the expected discounted reward. The three matrices (T, O, and R), mentioned earlier, are the elements that the proposed ontology-based generator extracts from the semantic knowledge graph and compiles into a model ready for solving.

2.2. System architecture

The system distinguishes semantic representation from numerical computation and provides an interface for an agent to handle planning and belief management. A high-level overview of how these components interact is shown in Fig. 1.

The architecture consists of two main layers:

- *the semantic layer:* implemented as an OWL 2 knowledge base hosted on an Apache Fuseki server (Apache Software Foundation, USA). It holds the domain-independent POMDP base ontology along with one or more domain ontologies, all stored as RDF triples and queried through SPARQL;
- *the computational layer:* made up of a POMDP model generator and a POMDP solver. The generator sends SPARQL queries to Fuseki, applies selection and aggregation policies to address conflicting claims, and exports a JSON description of the POMDP. The solver then takes this model and computes an approximately optimal policy using the SARSOP point-based solver (National University of Singapore, Singapore).

An orchestrating agent oversees the process, triggering model generation, initializing beliefs, and querying the solver for optimal actions based on the current belief state.

2.3. Experimental setup and benchmark generation

To empirically validate the compilation method, a diagnosis benchmark was constructed to generate custom test scenarios dynamically. The diagnosis ontology generator takes as parameters the number of components n_{comp}, the number of fault modes per component n_{mode}, bound $k_{\max}$ on the number of simultaneously faulty components. At a high level, the construction procedure is summarized in Fig. 2.

Combinatorial model size: Let $n_{comp} \geq 1$ denote the number of components and $n_{mode} \geq 1$ the number of fault modes per component. Component i can be in one of $n_{mode} + 1$ local modes $\{0, ..., n_{mode}\}$, where 0 represents a healthy component and $\{1, ..., n_{mode}\}$ are distinct fault modes. The generator restricts the number of simultaneously faulty components to at most $k_{\max}$. Hence, the maximal number of distinct fault states is

$$N_{fault}^{\max} = \sum_{k=1}^{k_{\max}} \binom{n_{comp}}{k} n_{mode}^{k}, \quad (1)$$

and the total number of states is

$$|S| = 2 + N_{fault}^{\max}, \quad (2)$$

where the constant 2 accounts for the goal and catastrophic terminal states. When a finite budget B is used, the actual number of non-terminal states is $min\{N_{fault}^{\max}, B\}$, but the qualitative scaling in n_{comp} and n_{mode} remains governed by (1).

The action set consists of two actions per component (inspection and repair) plus a *Wait* action, so that

$$|A| = 2n_{comp} + 1. \quad (3)$$

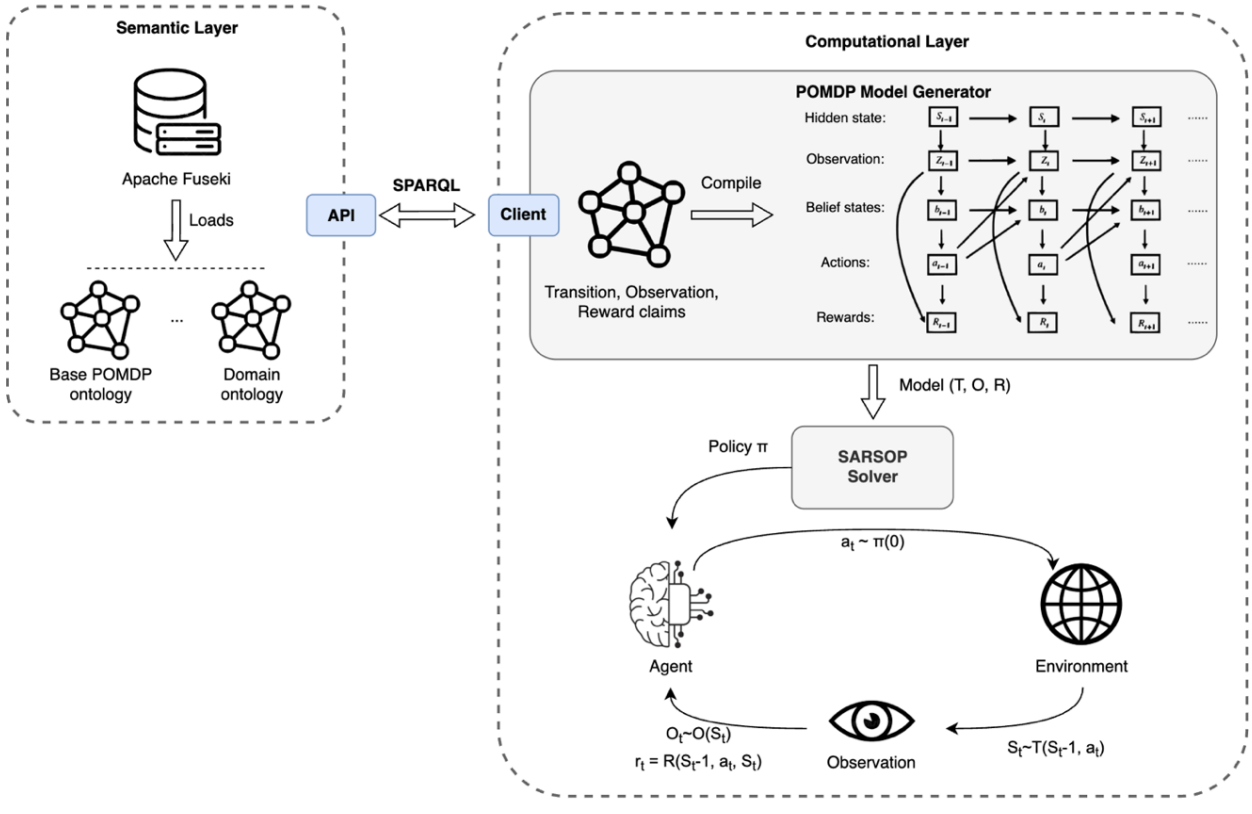

Fig. 1. Overall architecture of the proposed system

Algorithm 1: Diagnosis POMDP A-Box generation

Require: $n_{\text{comp}}, n_{\text{mode}}, k_{\max}$
1 create a *KnowledgeSource* individual for the generator
2 build state space:
  add goal (*State_Goal*) and catastrophic (*State_Fail*);
  add all single–fault patterns;
  for $2 \le k \le k_{\max}$, sample multi–fault patterns, ensuring that all subsets are present
3 build action set: for each component i, add *Probe_i* and *Repair_i*, plus *Wait*
4 build observation set: add three global observations and, for each (i, m), a fault signal
5 write *TransitionClaim* individuals for $(T(s' \mid s, a))$ using fixed templates for wait, probe and repair
6 write *PassiveObservationClaim* individuals for $O(o \mid s)$ with a dominant primary signal and noise
7 write *RewardClaim* and *CostClaim* individuals for state rewards and action costs

Fig. 2. Diagnosis POMDP A-Box generation algorithm

The observation space includes three global observations (healthy reading, catastrophic reading, noisy reading) and one fault-specific signal for every component and mode pair, hence

$$|\Omega| = 3 + n_{comp} n_{mode}. \quad (4)$$

These expressions allow each diagnosis instance to be characterized by the triple $(|S|, |A|, |\Omega|)$ as a direct function of $(n_{comp}, n_{mode}, k_{\max})$, and they explain the combinatorial growth patterns observed in the experiments results.

Benchmark datasets: With these parameters, two separate families of models were created to examine how various structural variables affect the solver's runtime and the system's overall scalability:

– *components scaling* (c_1–c_5): the number of components, n_{comp}, was gradually increased from 5 to 9 while maintaining fault modes fixed at $n_{mode} = 3$. This process simulates the addition of new physical parts to a diagnosed system. Importantly, as outlined by (3) and (4), increasing components simultaneously expands the state, action, and observation spaces, resulting in a more complex branching factor for the solver;

– *fault-modes scaling* (f_1–f_5): the components were fixed at $n_{comp} = 5$, but the number of fault modes n_{mode} per component was increased from 3 to 7. This simulates a scenario where the physical system remains unchanged, but diagnostic detail is enhanced. In this case, the action space A stays constant, allowing for evaluating the algorithm's performance in a densely populated state space without the complication of additional actions.

Each model was solved in offline mode using the SARSOP with a 600-second time limit and a target precision threshold of 0.05.

3. Results and Discussion

3.1. Domain-independent ontology schema for POMDP parameters

The base ontology provides a domain-independent T-Box for representing models under partial observability. At its core are three abstract classes, *SystemState*, *SystemAction* and *SystemObservation*, corresponding to the sets S, A and Ω respectively. Domain ontologies introduce specific objects as subclasses or instances of these abstract types.

To attach provenance and temporal information to probabilities and rewards, the ontology adopts an n-ary relation pattern represented as a class *KnowledgeClaim*. Each claim is an individual asserting a single parameter of the POMDP model. Claims are linked to their origin through the object property. Sources represent guidelines, textbooks, research papers, and may themselves carry trust scores or other metadata. The structure of this schema is summarized in Fig. 3.

ISSN-L 2664-9969; E-ISSN 2706-5448

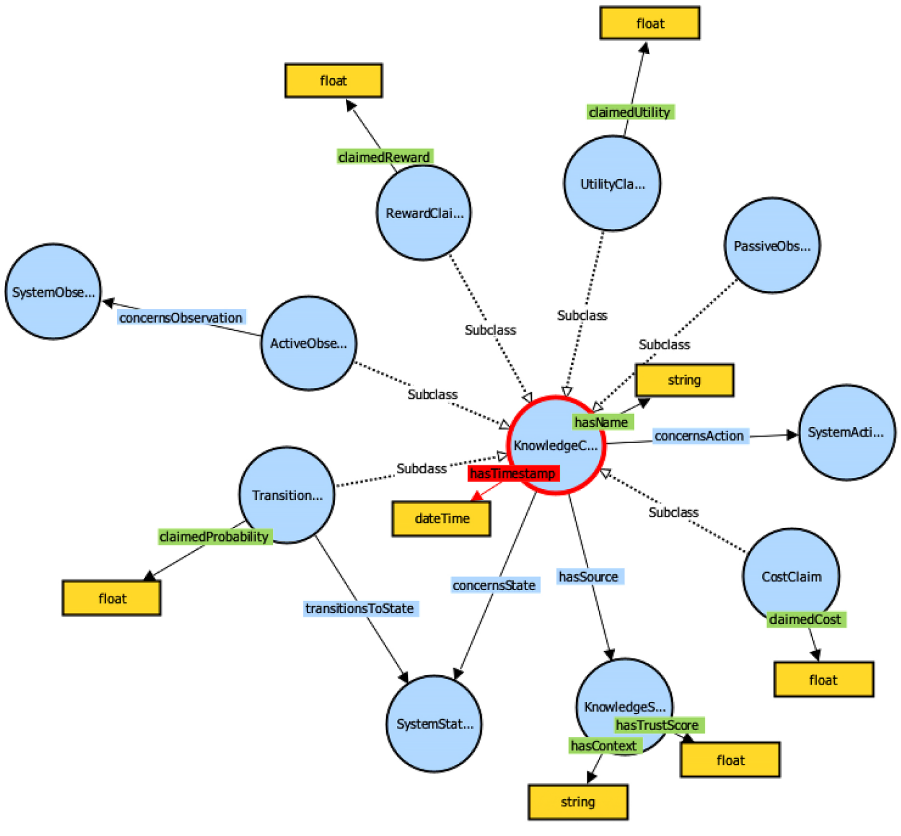

Fig. 3. Core structure of the base POMDP ontology. KnowledgeClaim acts as the central node; specialized subclasses (transition, observation, reward, cost, utility) link states, actions and observations to numerical values, timestamps and knowledge sources

Subclasses of *KnowledgeClaim* encode the different POMDP components as follows:

– TransitionClaim: links an action a, a start state and a successor state s' via the properties *concernsAction*, *concernsState* and *transitionsToState*. It assigns a numerical value through *claimedProbability*, thereby representing a single entry $T(s'|s, a)$;

– ObservationClaim: subdivided into *PassiveObservationClaim* and *ActiveObservationClaim*. These classes define $O(o|s)$ and $O(O|s',a)$ respectively, by relating states (and optionally actions) to observations;

– Reward and Cost Claims: rewards are expressed through *RewardClaim* for intrinsic state utility $R(s)$, *CostClaim* for action costs $R(a)$, and *UtilityClaim* for additional state-action terms $R(s, a)$, using properties such as *claimedReward* and *claimedCost*.

This design separates abstract definitions from instantiated claims, allowing the same base ontology to be reused for different domains with minimal changes.

3.2. Deterministic compilation procedure and conflict resolution

The generator module serves as a compiler from the RDF knowledge graph to a POMDP specification. It interacts with the triple store through abstraction that issues SPARQL queries. The output of the generator is a single model object comprising (I) the finite sets of states, actions and observations, and (II) transition function T, observation function O and reward function R in a form suitable for conversion into numerical tensors.

The extraction of the state, action and observation sets relies on the ontological subclass hierarchy. The generator selects all individuals that are instances of a class that is a (transitive) subclass of one of the abstract system classes *SystemState*, *SystemAction* or *SystemObservation*.

The construction of the transition function comprises of three stages:

Extraction. All instances of *TransitionClaim* are retrieved via SPARQL. Each retrieved claim is parsed into the tuple

$$\begin{pmatrix} a(c), s(c), s'(c), p(c), \\ t(c), src(c) \end{pmatrix}, \quad (5)$$

where $a(c) \in A$ – the associated action (*concernsAction*), $s(c) \in S$ the start state (*concernsState*), $s'(c) \in S$ the successor state (*transitionsToState*), $p(c) \in [0, 1]$ the asserted probability (*claimedProbability*), $t(c)$ the timestamp (*hasTimestamp*), and $src(c)$ the provenance source (*hasSource*).

Conflict resolution (policy-based selection/aggregation). Retrieved claims are grouped by the key (a, s, s'). For each key, let

$$C_{a,s,s'} = \\ = c : a(c) = a,\ s(c) = s,\ s'(c) = s', \quad (6)$$

denote the (multi)set of candidate claims for that parameter. Because $C_{a,s,s'}$ may contain multiple, potentially conflicting assertions (different sources, timestamps, contexts), the generator applies a *resolution policy* to produce a single compiled probability:

$$\pi : P(Claims) \to [0,1], \\ \hat{p}(a, s, s') = \pi(C_{a,s,s'}). \quad (7)$$

The policy π can be instantiated either as a *selection* policy (pick one claim and return its value) or as an *aggregation* policy (compute values from multiple claims).

Examples include:

1. *Latest-timestamp (used in experiments)*:

$$c^* = \arg\max_{c \in C_{a,s,s'}} t(c), \\ \hat{p}(a, s, s') = p(c^*) \quad (8)$$

2. *Trust-weighted aggregation*:

$$\hat{p}(a, s, s') = \sum_{c \in C_{a,s,s'}} w(c)\, p(c), \\ w(c) = \frac{\tau(src(c))}{\sum_{c' \in C_{a,s,s'}} \tau(src(c'))}, \quad (9)$$

where $\tau(\cdot)$ reads the source trust score.

3. *Source-priority selection:* choose c^* from the highest-priority source under a predefined ordering, breaking ties by timestamp, then set $\hat{p}(a, s, s') = p(c^*)$.

4. *Context match:* filter candidates to those which source context matches the deployment context, then apply either latest-timestamp or trust-weighted aggregation on the filtered set.

This paper presents results for the latest-timestamp policy aimed at isolating the scaling behavior of compilation and solving.

Normalization and defaulting. For each state and action pair (s, a), define the total outgoing mass to *non-self* successors

$$Z(s,a) = \sum_{u \in S,\ u \neq s} \hat{p}(a,s,u). \quad (10)$$

The compiled transition function is then defined by:

$$T(s'|s,a)=\begin{cases}1, & \text{if } Z(s,a)=0 \text{ and } s'=s,\\ 0, & \text{if } Z(s,a)=0 \text{ and } s'\neq s,\\ \hat{p}(a,s,s'), & \text{if } 0<Z(s,a)\leq 1 \text{ and } s'\neq s,\\ 1-Z(s,a), & \text{if } 0<Z(s,a)\leq 1 \text{ and } s'=s,\\ \dfrac{\hat{p}(a,s,s')}{Z(s,a)}, & \text{if } Z(s,a)>1 \text{ and } s'\neq s,\\ 0, & \text{if } Z(s,a)>1 \text{ and } s'=s.\end{cases} \quad (11)$$

Once the transition function has been instantiated, an analogous procedure is applied to construct the observation function. The generator first partitions the action set into *passive* and *active* actions. An action is classified as active if it appears as the value of *concernsAction* in at least one *ActiveObservationClaim*; all remaining actions are treated as passive.

For passive observations, all *PassiveObservationClaim* instances are grouped by state s. For each state, the associated values are normalised to obtain a probability distribution $O(o|s)$ over observations $o \in \Omega$. This state-conditioned distribution is then reused for every passive action a, so that

$$O(o|s,a)=O(o|s), \quad (12)$$

for all passive actions a.

For active observations, all *ActiveObservationClaim* instances are grouped by pairs (a, s'), and the corresponding values are normalized to define

$$O(o|s',a), \quad (13)$$

for each active action a and s'.

The reward function is derived from three types of claims. State rewards $R_s(s)$ are obtained from *RewardClaim* individuals, action costs $C(a)$ from *CostClaim*, and additional state-action utilities $U(s, a)$ from *UtilityClaim*. The resulting reward associated with a pair (s, a) is then computed as

$$R(s,a)=R_s(s)+U(s,a)-C(a), \quad (14)$$

resulting in a complete reward structure that is consistent with the ontological representation.

3.3. Benchmark evaluation on diagnosis POMDP model

The proposed ontology-based generation pipeline was evaluated on a family of automatically constructed diagnosis POMDPs. The experiments address three questions:

1) can proposed base ontology be used for scalable PODMP model instantiation;

2) how the size of the resulting model scales with the number of components and fault modes;

3) how this scaling affects offline solution time and policy quality for the SARSOP solver.

An illustrative fragment of generated model (using algorithm described in Fig. 2), projected onto the state space and transitions, is shown in Fig. 4. States correspond to fault patterns over components; green edges indicate successful repairs, grey edges self-loops and dashed edges rare catastrophic failures.

Table 1 summarizes the resulting model sizes and timings for 10 iterations of the benchmark configurations. Generation time is negligible compared with solution time: even for the largest instance (f_5 with almost 4000 states), the ontology file is created in under 4 seconds.

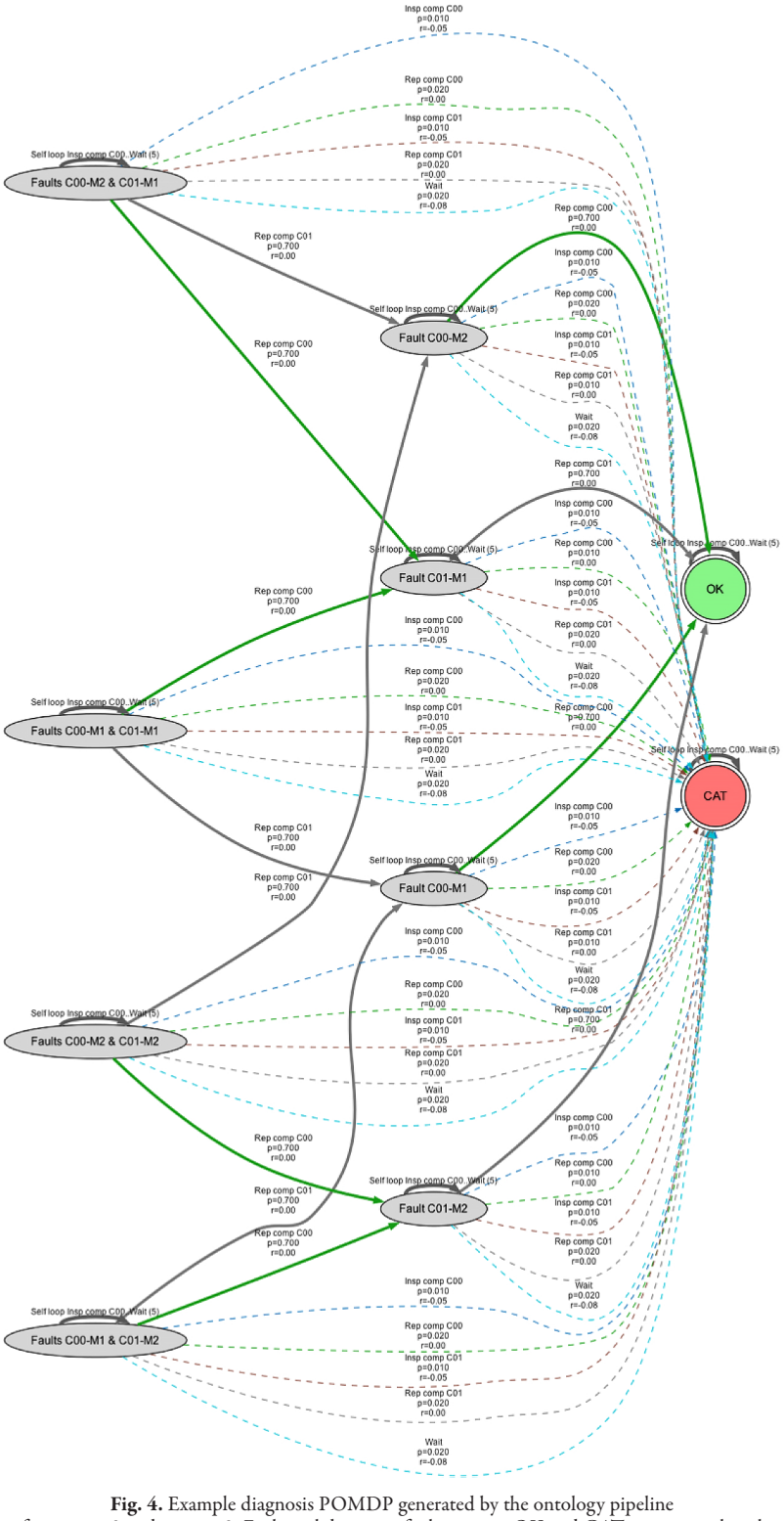

Fig. 4. Example diagnosis POMDP generated by the ontology pipeline for $n_{comp} = 2$ and $n_{mode} = 2$. Each oval denotes a fault pattern; OK and CAT are terminal goal and catastrophic states

ISSN-L 2664-9969; E-ISSN 2706-5448

Table 1

Diagnosis benchmark instances. $|C|$ denotes the number of components, $|F|$ the number of fault modes per component, $|S|$ the number of states, $|A|$ actions, $|\Omega|$ observations, Gen the ontology generation mean time, and Solve the SARSOP solve mean time

Instance	$\|C\|$	$\|F\|$	$\|S\|$	$\|A\|$	$\|\Omega\|$	Gen [s] (mean ± std)	Solve [s] (mean ± std)	Timeout rate
c_1	5	3	377	11	19	0.32 ± 0.08	4.23 ± 0.87	0.0
c_2	6	3	695	13	22	0.58 ± 0.01	44.01 ± 1.72	0.0
c_3	7	3	1157	15	25	1.14 ± 0.03	584.38 ± 46.02	0.5
c_4	8	3	1790	17	28	2.65 ± 0.48	630.92 ± 15.04	1.0
c_5	9	3	2621	19	31	3.59 ± 0.31	639.14 ± 8.98	1.0
f_1	5	3	377	11	19	0.28 ± 0.01	3.58 ± 0.19	0.0
f_2	5	4	822	11	24	0.60 ± 0.00	11.83 ± 0.56	0.0
f_3	5	5	1527	11	29	1.23 ± 0.13	33.66 ± 3.79	0.0
f_4	5	6	2552	11	34	2.04 ± 0.04	70.50 ± 8.33	0.0
f_5	5	7	3957	11	39	3.44 ± 0.01	144.40 ± 7.30	0.0

Resulting solve time and model size as a function of n_{comp} in the components scaling setting can be seen in Fig. 5. Specifically, Fig. 5, *a*, shows that SARSOP copes well with the two smallest instances, with solve times below 50 s. From c_2 to c_3, however, solve time jumps by an order of magnitude, and for c_4 and c_5 the runs effectively hit the 600 s limit. Fig. 5, *b*, illustrates that the number of states, actions and observations all grow with the number of components. This behavior is consistent with the exponential dependence of point-based solvers on the number of reachable belief states: as the underlying state space becomes denser and more branching, more belief points are needed to approximate the value function to the same precision.

Larger models require longer solution times to reach a comparable level of precision. Fig. 6 provides a complementary view by plotting the solver's precision. For the smallest models (c_1–c_2) the precision quickly drops below the target threshold of 0.05, whereas for c_4–c_5 it decreases slowly and does not fully stabilize before the time limit.

In the fault-modes scaling configuration, the number of components is fixed at $|C| = 5$, while the number of fault modes per component $|F|$ increases from 3 to 7. The resulting solve times and model sizes are shown in Fig. 7. Because $|A|$ remains constant (4), the growth in complexity is driven almost entirely by the state and observation spaces. Fig. 7, *b*, shows the empirical curves match the predicted quadratic-like growth of the model size, while Fig. 7, *a*, demonstrates the solve time. Unlike the components scaling experiment, all fault modes tests run well below the 600 s limit, with solve times under 150 s even at almost 4000 states. This difference can be explained by the structure of the state space: adding fault modes adds more states but does not introduce more actions, keeping the overall topology relatively similar, whereas adding components increases the dimensionality of the fault space and yields a much more complex transition structure.

The precision trajectories shown in Fig. 8 for f_1–f_5 decrease towards the target threshold, and even the largest model (f_5) reaches acceptable precision within the limit time.

To compare the effect of model size from the specific scaling parameter, Fig. 9 plots solve time against the number of states. For the same number of states, instances obtained by increasing fault modes (dashed line) are systematically easier to solve than instances of comparable size obtained by increasing components (solid line).

This observation highlights a significant benefit of the ontology-based approach: because the model is created from a structured description of components and fault modes, it allows for systematic exploration of families of instances with similar structure but different sizes. The results show that, for this specific solver and reward structure, adding more components is more challenging than adding more detailed fault categories, even when the number of states is similar.

The diagnosis benchmark is mainly used as a controlled method for generating POMDPs to confirm that complete models can be built from an ontology and reliably recompiled as parameters increase. Within the tested range (up to $|S| = 3957$), ontology generation took only seconds (Table 1). This finding addresses a common challenge in hybrid semantic-planning systems like CORPP [5] and ontology-driven observation frameworks, where semantic knowledge typically guides constraints or partial model components, while the full parameterization is usually external or manually crafted [7, 8].

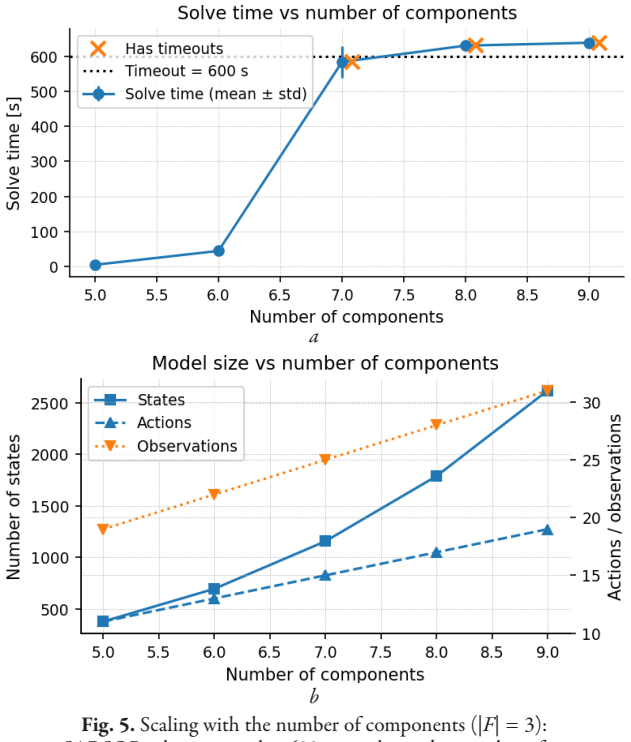

Fig. 5. Scaling with the number of components ($|F| = 3$): *a* – SARSOP solve time with a 600 s time limit; *b* – number of states, actions and observations

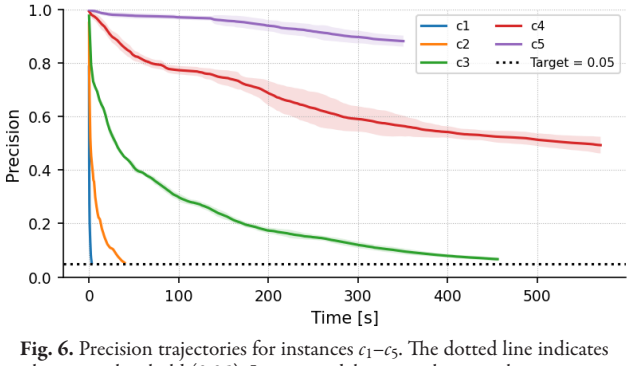

Fig. 6. Precision trajectories for instances c_1–c_5. The dotted line indicates the target threshold (0.05). Larger models require longer solution times to reach a comparable level of precision

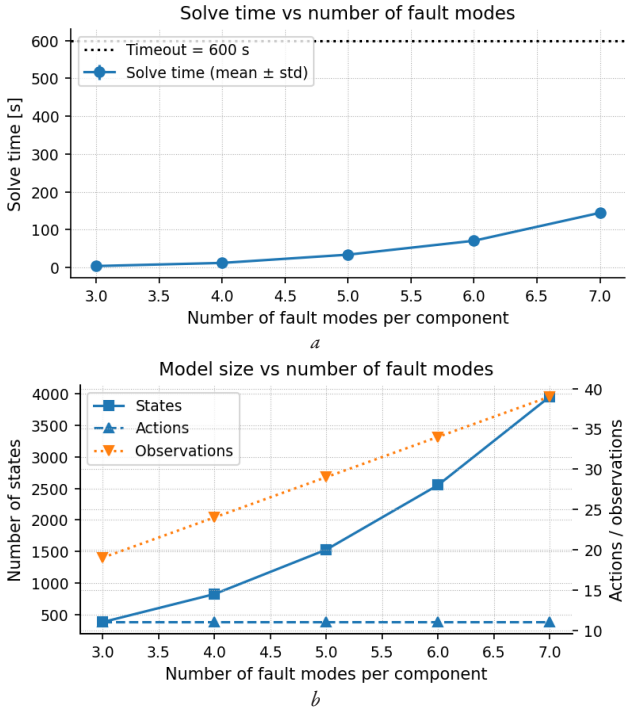

Fig. 7. Scaling with the number of fault modes ($|C| = 5$): *a* – SARSOP solve time; *b* – number of states, actions and observations

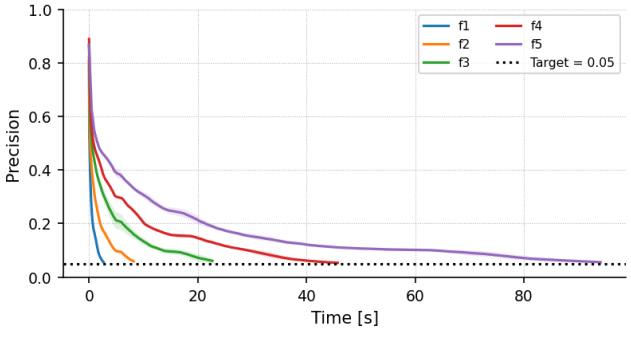

Fig. 8. Precision trajectories for instances f_1–f_5. Increasing the number of fault modes leads to slower convergence, but all runs reach the target precision before the time limit

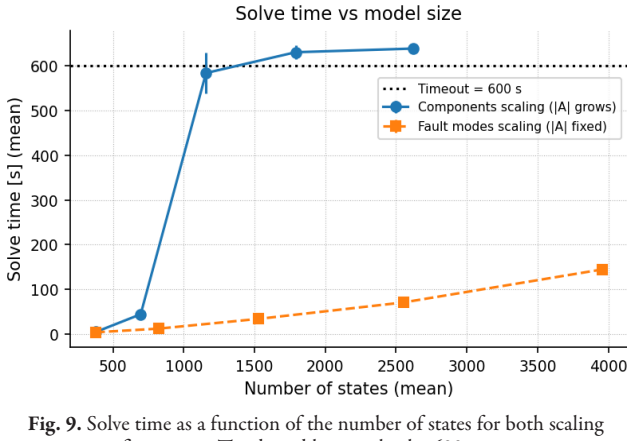

Fig. 9. Solve time as a function of the number of states for both scaling configurations. The dotted line marks the 600s timeout

The provenance claim design represents the knowledge graph as a manageable parameter store aligned with standard practices. PROV-O provides a universal provenance vocabulary for the Web [14]. Nanopublications offer a format for publishing individual RDF assertions with provenance and metadata [15, 16]. This paper's contribution is to adapt this approach to POMDP parameters using an *n*-ary claim pattern and link it to a deterministic compilation process that generates *T*, *O*, and *R* matrices. Unlike probabilistic ontology frameworks like BayesOWL [10], PR-OWL [11], and DISPONTE [12], which mainly enable probabilistic reasoning within the semantic layer, and unlike deterministic ontology-to-PDDL compilation pipelines [17–19].

3.4. Limitations and future work

The empirical evaluation is limited to the diagnosis benchmark family used in this research, so its external validity to other domains depends on how well their ontologies can be aligned with the required structure of state, action, and observation. The findings primarily address solver scalability and correctness of compilation, without assessing online performance in real-world deployments where delayed feedback could influence results.

Future work aims to enhance the compilation layer by incorporating more sophisticated conflict-resolution strategies (such as trust or context-aware methods), supporting incremental recompilation during knowledge updates, and enabling online operation where an agent can selectively expand the ontology with new claims based on their anticipated impact on the policy.

4. Conclusions

1. A domain-independent ontology schema was developed to represent numeric POMDP parameters as provenance-aware claims. As a qualitative outcome, the schema effectively models transitions, observations, rewards, and costs as reified entities with explicit metadata, including information sources, timestamps, and context. This approach shifts POMDP engineering from error-prone manual editing of numeric tables to managing fully auditable knowledge artifacts. It facilitates faster model updates and supports the integration of ontology learning pipelines, all while maintaining data governance and accountability.

2. A deterministic compilation process was designed to extract claims, resolve conflicts, and normalize parameters. This process uses SPARQL queries to identify semantic individuals, applies configurable policies to address conflicting assertions, and performs mathematical normalization to produce comprehensive, valid output tables for transition (T), observation (O), and reward (R) matrices. The main advantage is an entirely automated, reproducible conversion from a semantic knowledge graph into the precise formats needed by standard POMDP solvers, eliminating the need for human intervention.

3. A benchmark evaluation of the proposed method was conducted using an automated diagnosis model to evaluate generation speed and solver scalability. The experiments showed that ontology generation is highly efficient, taking less than 3.6 seconds even for the largest models (up to $|S| = 3957$). Solution times with the SARSOP solver varied predictably based on the scaling mode, ranging from 3.58 ± 0.19 seconds for baseline models to a 600-second timeout for more complex models. The analysis demonstrated that increasing structure by adding physical components – thus expanding action/observation spaces and transition branches – is more computationally intensive and results in different solve-time profiles compared to scaling only by fault modes, even when the total number of states is similar.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

Financing

The research was performed without financial support.

ISSN-L 2664-9969; E-ISSN 2706-5448

Data availability

Data will be provided upon request.

Use of artificial intelligence

Model: ChatGPT 5.2.

Sections: Introduction, Summary, and Related Work.

Use: The AI tool was utilized for structural revision and text rephrasing to increase readability and clarity.

Verification: The authors verified all AI-recommended changes and made sure the text is accurate and maintains its original meaning.

Impact: AI tool's output didn't affect neither the research data nor the research final conclusions.

Authors' contributions

Yaroslav Teplyi: Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization; ***Dmytro Dosyn***: Conceptualization, Supervision, Project administration, Writing – review and editing.

References

1. Musen, M. A., Noy, N. F., Shah, N. H., Whetzel, P. L., Chute, C. G., Story, M.-A., Smith, B. et al. (2012). The National Center for Biomedical Ontology. *Journal of the American Medical Informatics Association, 19 (2),* 190–195. https://doi.org/10.1136/amiajnl-2011-000523
2. Hoehndorf, R., Dumontier, M., Gkoutos, G. V. (2012). Evaluation of research in biomedical ontologies. *Briefings in Bioinformatics, 14 (6),* 696–712. https://doi.org/10.1093/bib/bbs053
3. Beard, R. W. (2018). Decision Making Under Uncertainty: Theory and Application. *IEEE Control Systems, 38 (6),* 114–115. https://doi.org/10.1109/mcs.2018.2866656
4. Iocchi, L., Lukasiewicz, T., Nardi, D., Rosati, R. (2009). Reasoning about actions with sensing under qualitative and probabilistic uncertainty. *ACM Transactions on Computational Logic, 10 (1),* 1–41. https://doi.org/10.1145/1459010.1459015
5. Zhang, S., Stone, P. (2015). CORPP: Commonsense Reasoning and Probabilistic Planning, as Applied to Dialog with a Mobile Robot. *Proceedings of the AAAI Conference on Artificial Intelligence, 29 (1).* https://doi.org/10.1609/aaai.v29i1.9385
6. Amiri, S., Shokrolah Shirazi, M., Zhang, S. (2020). Learning and Reasoning for Robot Sequential Decision Making under Uncertainty. *Proceedings of the AAAI Conference on Artificial Intelligence, 34 (3),* 2726–2733. https://doi.org/10.1609/aaai.v34i03.5659
7. Moulouel, K., Chibani, A., Amirat, Y. (2023). Ontology-based hybrid commonsense reasoning framework for handling context abnormalities in uncertain and partially observable environments. *Information Sciences, 631,* 468–486. https://doi.org/10.1016/j.ins.2023.02.078
8. Ghanadbashi, S., Zarchini, A., Golpayegani, F. (2023). An Ontology-Based Augmented Observation for Decision-Making in Partially Observable Environments. *Proceedings of the 15th International Conference on Agents and Artificial Intelligence.* Lisbon, 343–354. https://doi.org/10.5220/0011793200003393
9. Golpayegani, F., Ghanadbashi, S., Zarchini, A. (2024). Advancing Sustainable Manufacturing: Reinforcement Learning with Adaptive Reward Machine Using an Ontology-Based Approach. *Sustainability, 16 (14),* 5873. https://doi.org/10.3390/su16145873
10. Ding, Z., Peng, Y., Pan, R. (2006). BayesOWL: Uncertainty Modeling in Semantic Web Ontologies. *Soft Computing in Ontologies and Semantic Web.* Springer, 3–29. https://doi.org/10.1007/978-3-540-33473-6_1
11. Carvalho, R. N., Laskey, K. B., Costa, P. C. G. (2017). PR-OWL – a language for defining probabilistic ontologies. *International Journal of Approximate Reasoning, 91,* 56–79. https://doi.org/10.1016/j.ijar.2017.08.011
12. Bellodi, E., Lamma, E., Riguzzi, F., Albani, S. (2011). A distribution semantics for probabilistic ontologies. *URSW, 778,* 75–86. Available at: https://dl.acm.org/doi/10.5555/2887702.2887709
13. Lukasiewicz, T. (2008). Probabilistic description logic programs under inheritance with overriding for the Semantic Web. *International Journal of Approximate Reasoning, 49 (1),* 18–34. https://doi.org/10.1016/j.ijar.2007.08.005
14. Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J.; Lebo, T., Sahoo, S., McGuinness, D. (Eds.) (2013). *PROV-O: The PROV ontology.* Available at: https://www.w3.org/TR/prov-o/
15. Kuhn, T., Chichester, C., Krauthammer, M., Queralt-Rosinach, N., Verborgh, R., Giannakopoulos, G. et al. (2016). Decentralized provenance-aware publishing with nanopublications. *PeerJ Preprints.* https://doi.org/10.7287/peerj.preprints.1760v1
16. Kuhn, T., Banda, J. M., Willighagen, E., Ehrhart, F., Evelo, C., Malas, T. B. et al. (2018). Nanopublications: A Growing Resource of Provenance-Centric Scientific Linked Data. *2018 IEEE 14th International Conference on E-Science (E-Science),* 83–92. https://doi.org/10.1109/escience.2018.00024
17. Klusch, M., Gerber, A., Schmidt, M. (2005). Semantic web service composition planning with OWLS-XPlan. *AAAI Fall Symposium: Agents and the Semantic Web,* 55–62. Available at: https://www.researchgate.net/publication/228711042_Semantic_Web_service_composition_planning_with_OWLS-XPlan
18. John, T., Koopmann, P. (2023). Towards ontology-mediated planning with OWL DL ontologies. *CEUR Workshop Proceedings.* https://doi.org/10.48550/arXiv.2308.08200
19. Malburg, L., Klein, P., Bergmann, R. (2023). Converting semantic web services into formal planning domain descriptions to enable manufacturing process planning and scheduling in industry 4.0. *Engineering Applications of Artificial Intelligence, 126,* 106727. https://doi.org/10.1016/j.engappai.2023.106727

✉***Yaroslav Teplyi,*** *PhD Student, Department of Information Systems and Networks, Lviv Polytechnic National University, Lviv, Ukraine, ORCID: https://orcid.org/0009-0001-5548-5530, e-mail: yaroslav.b.teplyi@lpnu.ua*

Dmytro Dosyn, *Doctor of Technical Sciences, Professor, Head of Department of Information Systems and Networks, Lviv Polytechnic National University, Lviv, Ukraine, ORCID: https://orcid.org/0000-0003-4040-4467*

✉*Corresponding author*