# Aligning Early and Late Requirements Through Ontologies

Ricardo Ramalho and Juliana B.S. França[Supervisor]

Instituto de Computação, Universidade Federal do Rio de Janeiro, RJ, Brasil
{ricardorlj,julianabsf}@ic.ufrj.com

**Abstract.** The misalignment between early and late requirements in software development represents a significant challenge, directly impacting the quality of the final software product. The inconsistency between the stakeholders' initial vision and the technical implementation can result in systems that do not meet customer expectations, leading to rework, increased costs, and dissatisfaction. Early requirements represent the stakeholders' needs, while late requirements are refined and formalized to guide technical development. The transition between these phases is critical, and misalignments can lead to misinterpretations and inadequate final products. To mitigate this issue, the study proposes the use of ontologies as a tool to structure and formalize the knowledge related to requirements, facilitating project communication and documentation. Aspects of the main ontology development methodologies are analyzed, and the reuse of an existing ontology that fits the purposes of this study is addressed. This research presents the formalization of these concepts, indicating that the application of ontologies in a project's early requirements can positively contribute to the construction of systems adherent to stakeholder needs. As a future perspective, the evolution of the suggested adaptations into a formalized ontology is proposed, along with the introduction of goal-oriented approaches.

**Keywords:** Ontology, Requirements Engineering, Early Requirements, Late Requirements.

## 1 Introduction

Requirements are the foundation of any project, defining what stakeholders need from a potentially new system and what the system must do to meet that need. With the increasing complexity of software development, the importance of effective requirements engineering becomes evident [8]. In large software projects, requirements must initially be defined abstractly to avoid predefined solutions and allow competition among suppliers. After contracting, a detailed system definition must be developed for client validation, integrating both phases into a single system requirements document [6].

This divergence in the interpretation of what constitutes a requirement can lead to misunderstandings and inconsistencies during the evolution of requirements throughout the engineering process. A definition of requirements must

be broad and take into account various factors present in this complex process. Among these factors are the needs related to the requirements, which must be met by the product built based on them. The term "requirement" can also apply to the context in which real characteristics are the object of evaluation of a concrete product, based on conditions that this product must meet. Another way to view requirements is through the construction of a document that materializes the desired system features; in this sense, a requirement is a specification present in the document [27].

All three aspects of the requirements definition mentioned above are individually characterized in ISO/IEC/IEEE 24765:2017 [15]. However, this standard also expresses a fourth definition, which encompasses and complements these three aspects: requirement is a condition or capability that must be met or possessed by a system, system component, product, or service to satisfy an agreement, standard, specification, or other formally imposed documents. Requirements include the quantified and documented needs, wants, and expectations of the sponsor, customer, and other stakeholders.

In this context, the research proposes the use of ontologies as a tool to mitigate the misalignment between early and late requirements. The research is structured as follows: Section 2 discusses works related to the present research, involving studies that address early and late requirements and ontological approaches that operate on these phases. Sections 3 and 4 discuss the concepts of early and late requirements, and then the challenge of their misalignment is highlighted. Section 5 introduces the concept of ontologies and their application in Software Engineering, showing how they can be used to structure knowledge related to requirements. We then propose and exemplify the reuse of an ontology created to assist requirements traceability in the context of agile development.

In summary, this research addresses a critical problem in software development—requirements misalignment—and proposes an ontology-based solution. By structuring and formalizing knowledge related to requirements, ontologies can play a crucial role in ensuring that developed systems fully meet stakeholders' needs, reducing costs, and improving the quality of the final product.

## 2 Related Work

Vingerhoets et al. [28] explored the use of i* and UML in Blockchain-Oriented Software Engineering, showing that i* is effective for early requirements and UML for late requirements. While not ontological, their work highlights the importance of distinguishing and managing the transition between requirements phases—a challenge this study addresses using ontologies.

Singh et al. [24] addressed the transition between early and late requirements in the context of Data Warehouse development, highlighting the lack of clear distinction in existing approaches. They proposed an agent-oriented extension of the GDI (Goal-Decision- Information) model to bridge these phases. Unlike their domain-specific method, this study adopts a formal and generic approach using ontologies to align early and late requirements.

Fernandes et al. [11] proposed using the Tropos methodology to model Competency Questions (CQs) in the early stages of ontology engineering, drawing parallels between CQs and system requirements. Their approach emphasizes the use of early and late requirements phases to connect organizational objectives to ontology scope, reinforcing the value of methodological rigor and phase distinction—key concerns also addressed in this study.

## 3 Early Requirements

Before drafting any requirements, it is necessary to understand the **whys** that justify the system under development [31]. In this context, early requirements emerge, connecting the proposed system to organizational goals, its purpose, available alternatives, and the implications of these choices for stakeholders [24].

The early requirements approach involves identifying all relevant agents in the system and their respective goals. The result of this process should be an organizational model that represents these pairs [4] [10]. Recently, Kadakolmath and Ramu [17] described early requirements as follows:

> Early-phase requirements focus on the WHY dimension of requirements engineering. That is, they focus on modeling and analysis of the environment of the software system. Also, they illustrate how stakeholders' needs, motivations, and complex relationships may be addressed by multiple alternatives.

Since this is an initial stage, the information is often in a raw state, requiring greater analytical effort. With the evolution of software development, specific tools have been developed to address this challenge, many of which focus on graphical support to facilitate the visualization of information collected in the early stages of projects [4, 9, 17].

Early requirements can thus be defined as the set of interests and needs of the system's stakeholders. Through continuous interaction with these stakeholders, requirements are elicited, recorded, and refined until they are ready to form a document that fully reflects these interests. After this stage, the transition to the final requirements document takes place, serving as the foundation for project development. During this phase, the requirements—now referred to as late requirements—are detailed and organized to ensure clarity for the development team, ensuring that the requesters' needs are properly understood and met.

## 4 Late Requirements

In an ideal scenario, the artifact generated from stakeholder interaction would be sufficient to initiate project development. However, in practice, this reality is significantly more complex and challenging. The misalignment of objectives among stakeholders makes negotiation indispensable for handling contradictory requirements [1]. During this process, the goal is to draft a requirements document in

natural language to facilitate communication. However, this approach can introduce ambiguities, allowing different actors to interpret the same requirement in different ways.

To mitigate the problems arising from this initial stage, Lamsweerde [18] proposes a specification and documentation phase, which follows a structured process, represented in Figure 1:

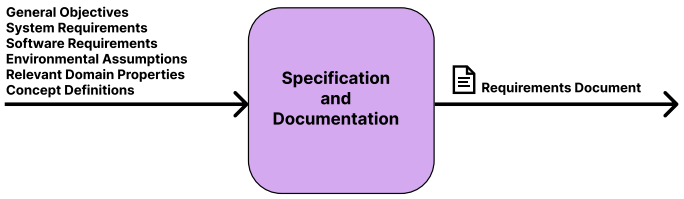


**Fig. 1.** Specification and Documentation Process

The Requirements Document, the main artifact resulting from this stage, presents a coherent structure designed to precisely specify the early requirements. This specification eliminates ambiguities, limits the project scope, and serves as the foundation for technical development. In this context, late requirements emerge as a refined and structured set of specifications, documented to provide clear instructions to those responsible for the project's development.

A project developed based on the Requirements Document should fully meet the initial interests and desires of stakeholders. To achieve this, it is essential to avoid distortions between early and late requirements, ensuring a smooth transition between these phases. This alignment between the two types of requirements has been widely studied in recent years in the field of software engineering [28].

## 5 Requirements Misalignment

Although the roles of the early and late phases in requirements engineering seem well defined, the transition between these stages has been widely debated over the past three decades in software development. This process is challenging because early requirements, often expressed ambiguously and informally, need to be refined and formalized to serve as a solid foundation for the system's technical development [32]. An illustration of the potential impact of this misalignment is presented by Young [30] in the following scenario:

> It's your worst nightmare. A customer walks into your office, sits down, looks you straight in the eye, and says, "I know you think you understand what I said, but what you don't understand is what I said is not what I meant." Invariably, this happens late in the project, after deadline

commitments have been made, reputations are on the line, and serious money is at stake.

Problems arising from requirements misalignment, when identified, often require corrections, the cost of which varies depending on the stage of development in which they are detected [23].

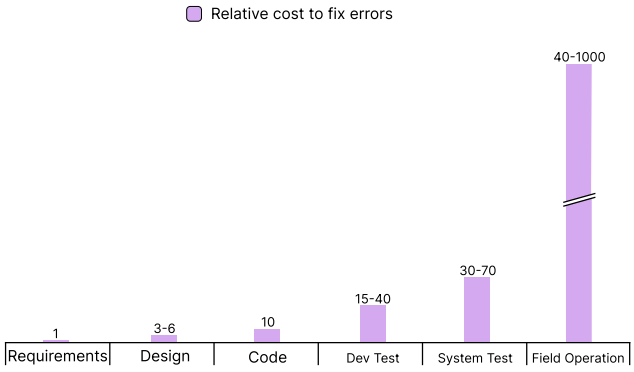


**Fig. 2.** Relative cost of error correction by software development stage [3], adapted.

An analysis of Figure 2 reveals the exponential increase in costs associated with fixing errors as the project progresses. This observation reinforces the importance of effective alignment between early and late requirements. To ensure that stakeholder interests are adequately addressed, it is crucial that early requirements are fully incorporated into late requirements. Recent literature demonstrates that the problem still persists, causing damage to the industry [17,19,26,28]. Furthermore, it is essential to prevent the introduction of late requirements that are not linked to an initial requirement, thereby promoting a more objective and efficient project.

Thus, a tool that enables effective alignment between early and late requirements could ensure that these conditions are met, minimizing costs and maximizing the quality of the final product.

## 6 Ontologies

Conceptual modeling was recognized as important for the development of information systems as early as the 1960s [29]. Serving as an explicit specification of conceptualization [13], ontologies have been a growing area of interest in information science research [2]. Ontology is a term originating from philosophy that defines a systematic representation of existence. In an ontology, a universe

can be explicitly represented by entities, their relationships, and governing rules, which makes the represented universe a measurable and more easily analyzable object [13].

In the 1990s, Guarino [14] introduced the term "Ontology-Driven Information Systems" to describe information systems that formally use ontologies. Later, Jurisica et al. [16] proposed using ontologies for knowledge management, highlighting their benefits in structuring knowledge and improving interoperability between systems. Ontologies help standardize concepts, reduce subjectivity, and improve communication. Beyond interoperability, they also enhance user-database interactions by enabling more intuitive and user-focused searches [21].

### 6.1 Ontology for requirements alignment

As discussed in Sections 3 and 5, misalignment arises from poor management of abstract initial requirements as they evolve into late requirements. Ontologies support the formalization and structuring of requirements, enabling analysis of their relationships to identify unmet initial demands or unexpected late-stage additions, thus helping to reduce misalignment.

The process of building ontologies in software engineering has proven to be more of an artistic endeavor than a methodological one [25]. A study conducted by Cardoso in 2007 [5], showed that more than half of the participants did not use any methodology. By using a methodology focused on ontology construction, common consistency errors in practice—such as missing or overlapping concepts, maintenance difficulties, and lack of standardization—are mitigated. Analyzing the methodologies present in previous studies [5] [25] [12], we observe that the reuse analysis step is present in several methodologies. Aiming to enhance requirements traceability in agile software development, Murtazina and Avdeenko [22] presented an ontology-based approach. They developed a formal knowledge representation using the Web Ontology Language (OWL) to facilitate this traceability.

OWL (Web Ontology Language) is a description logic-based language used to formally represent knowledge in ontologies, enabling automatic inference and consistency checking [20]. It has been widely applied in requirements engineering, including in half of the 66 studies reviewed by Dermeval et al. [7]. In this study, OWL is suitable for aligning early and late requirements, as it supports the formal definition of requirement types and relationships. Its compatibility with reasoners like HermiT allows for the inference of refinement and traceability links, aiding in the detection of inconsistencies and alignment gaps.

## 7 Reusing and Extending Ontology

### 7.1 Reuse Justification

The ontology proposed by Murtazina and Avdeenko is suitable for the present study for several technical and conceptual reasons that meet the objective of

aligning early and late requirements through a formal ontological representation. The ontology is built in OWL and implemented in the Protégé tool with support for the previously mentioned HermiT reasoner, which allows not only the formal definition of concepts and relations but also the automatic inference of connections between requirements and artifacts. This capability is essential in a study whose focus is to establish and verify semantic coherence between requirements from different phases.

Furthermore, the analyzed ontology already includes a rich structure of concepts relevant to requirements engineering, such as *Requirement*, *RequirementsArtefact*, *Stakeholder*, *TestType*, *Task*, *ProductFeature*, among others, which reduces the effort of modeling from scratch and allows for the reuse and extension of existing concepts. The ontology also introduces properties like *traceFrom* and *traceTo*, with subproperties such as refines, *isPartOf*, *isTestedBy*, and *hasSource*, which are directly applicable to the task of aligning early requirements (e.g., strategic objectives or stakeholder intentions) with late requirements (such as specific functionalities or acceptance criteria. This alignment can be evaluated using metrics that assess the extent to which early requirements are reflected in late requirements, as well as the proportion of late requirements that are demonstrably derived from early-stage specifications.

## 8 Example Scenario

To illustrate the reuse and extension of the ontology proposed by Murtazina and Avdeenko [22], we present a practical scenario involving the development of a *Course Management System (CMS)* for a university. This system aims to support professors, students, and administrators in managing courses, enrollments, and assessments.

### 8.1 Early and Late Requirements

During the elicitation phase, stakeholders expressed several high-level goals, which we classify as early requirements (ERs) in Section 3:

- **ER1**: Reduce manual effort in course registration.
- **ER2**: Enable professors to manage course content and grades easily.
- **ER3**: Improve student satisfaction through better access to course information.

From these, we derive late requirements (LRs) that represent concrete system functionalities:

- **LR1** (from ER1): The system shall allow students to register for courses online via a web interface.
- **LR2** (from ER2): Professors shall be able to upload course materials and input grades into the system.
- **LR3** (from ER3): Students shall receive automatic notifications when course materials are updated.

### 8.2 Ontology Reuse and Extension

The original ontology already provides a foundation with the `Requirement` class and properties such as `traceFrom`, `traceTo`, and `refines`, which are suitable for establishing traceability. To adapt the ontology for aligning early and late requirements, we sugest the introduction of the following extensions:

- Subclasses of `Requirement`: `EarlyRequirement` and `LateRequirement`.
- A new object property: `isDerivedFrom`, to link `LateRequirement` instances to their corresponding `EarlyRequirement`.
- A datatype property: `requirementPhase`, with values `"early"` and `"late"`, to annotate the abstraction level.

### 8.3 Formal Representation in OWL

Below is a simplified fragment of how the requirements and their relationships can be represented in OWL syntax:

```
:ER1 rdf:type :EarlyRequirement ;
     :description "Reduce manual effort in course registration" .
:LR1 rdf:type :LateRequirement ;
     :description "Enable online student registration via web interface" ;
     :isDerivedFrom :ER1 .
:requirementPhase a owl:DatatypeProperty ;
     rdfs:domain :Requirement ;
     rdfs:range xsd:string .
:ER1 :requirementPhase "early" .
:LR1 :requirementPhase "late" .
```

This example demonstrates how reusing and extending an existing ontology facilitates the semantic alignment of requirements across abstraction levels, while maintaining the benefits of formal reasoning and traceability in software engineering.

## 9 Conclusion

This study addressed the challenges of requirements misalignment, emphasizing how poor communication can lead to failures in the final product. Ontologies emerge as a promising alternative for structuring and formalizing knowledge in Requirements Engineering, promoting traceability and coherence. The reuse of the ontology by Murtazina and Avdeenko was suggested for constructing an ontology that addresses this problem, with the necessary modifications to ensure alignment between requirements.

The use of ontologies enables the creation of more precise conceptual models, facilitating the integration between early and late requirements. This contributes to reducing ambiguities and improving communication among stakeholders. While the reuse and extension of the selected ontology shows promise,

further validation through case studies or tool implementation is needed to assess scalability and real-world applicability. As future work, we propose the development of an ontology capable of establishing links between early and late requirements, while also incorporating goal-oriented approaches, such as i* and Tropos.

## References


1. Ahmad, S.: Negotiation in the requirements elicitation and analysis process. In: 19th Australian Conference on Software Engineering (aswec 2008). pp. 683–689 (2008)
2. Almeida, M.B., de Oliveira, V.N.P., Coelho, K.C.: Estudo exploratório sobre ontologias aplicadas a modelos de sistemas de informação: perspectivas de pesquisa em ciência da informação. Encontros Bibli: revista eletrônica de biblioteconomia e ciência da informação **15**(30), 32–56 (2010)
3. Boehm, B.W.: Software engineering economics. Springer (2002)
4. Brinkkemper, J., Solvberg, A.: Tropos: A framework for requirements-driven software development. Info. Syst. Engg.: State Art Res. Themes **1**, 261 (2000)
5. Cardoso, J.: The semantic web vision: Where are we? IEEE Intelligent systems **22**(5), 84–88 (2007)
6. Davis, A.M.: Software requirements: objects, functions, and states. Prentice-Hall, Inc. (1993)
7. Dermeval, D., Vilela, J., Bittencourt, I.I., Castro, J., Isotani, S., Brito, P.: A systematic review on the use of ontologies in requirements engineering. In: 2014 brazilian symposium on software engineering. pp. 1–10. IEEE (2014)
8. Dick, J., Hull, E., Jackson, K.: Requirements engineering. Springer (2017)
9. Dubois, E., Yu, E., Petit, M.: From early to late formal requirements: a process-control case study. In: Proceedings Ninth International Workshop on Software Specification and Design. pp. 34–42 (1998)
10. ElSayed, I.A., Ezz, Z., Nasr, E.: Goal modeling techniques in requirements engineering: A systematic literature review. J. Comput. Sci. **13**(9), 430–439 (2017)
11. Fernandes, P.C.B., Guizzardi, R.S., Guizzardi, G.: Using goal modeling to capture competency questions in ontology-based systems. Journal of Information and Data Management **2**(3), 527–527 (2011)
12. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. Springer Science & Business Media (2006)
13. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition **5**(2), 199–220 (1993)
14. Guarino, N.: Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy, vol. 46. IOS press (1998)
15. ISO/IEC/IEEE: ISO/IEC/IEEE 24765:2017 - Systems and software engineering – Vocabulary (Sep 2017), available at: https://www.iso.org/standard/71952.html
16. Jurisica, I., Mylopoulos, J., Yu, E.: Ontologies for knowledge management: an information systems perspective. Knowledge and Information systems **6**, 380–401 (2004)
17. Kadakolmath, L., Ramu, U.D.: istar goal model to z formal model translation and model checking of cbtc moving block interlocking system. Form. Asp. Comput. **36**(1) (2024)

18. van Lamsweerde, A.: Requirements Engineering. Wiley (2009)
19. Lewellen, S.: A comprehensive approach to identifying key stakeholders in complicated software ecosystems. In: 2021 IEEE 29th International Requirements Engineering Conference (RE). pp. 492–497 (2021)
20. McGuinness, D.L., Van Harmelen, F., et al.: Owl web ontology language overview. W3C recommendation **10**(10), 2004 (2004)
21. Munir, K., Anjum, M.S.: The use of ontologies for effective knowledge modelling and information retrieval. Applied Computing and Informatics **14**(2), 116–126 (2018)
22. Murtazina, M.S., Avdeenko, T.: An ontology-based approach to support for requirements traceability in agile development. Procedia Computer Science **150**, 628–635 (2019)
23. Rothman, J.: What does it cost you to fix a defect? and why should you care. Retrieved March **13**, 2010 (2000)
24. Singh, Y., Gosain, A., Kumar, M.: From early requirements to late requirements modeling for a data warehouse. In: 2009 Fifth International Joint Conference on INC, IMS and IDC. pp. 798–804 (2009)
25. Soares, A.: Towards Ontology-Driven Information Systems: Guidelines to the Creation of New Methodologies to Build Ontologies. Ph.D. thesis, The Pennsylvania State University (2009)
26. Unterkalmsteiner, M., Abrahamsson, P., Wang, X., Nguyen-Duc, A., Shah, S.M.A., Bajwa, S.S., Baltes, G.H., Conboy, K., Cullina, E., Dennehy, D., et al.: Software startups–a research agenda. arXiv preprint arXiv:2308.12816 (2023)
27. Vazquez, C.E., Simões, G.S.: Engenharia de Requisitos: software orientado ao negócio. Brasport (2016)
28. Vingerhoets, A., Heng, S., Wautelet, Y.: Using i* and uml for blockchain oriented software engineering: Strengths,weaknesses, lacks and complementarity. Complex Systems Informatics and Modeling Quarterly **0**, 26–45 (2021)
29. Wand, Y., Weber, R.: Research commentary: information systems and conceptual modeling—a research agenda. Information systems research **13**(4), 363–376 (2002)
30. Young, R.R.: Effective requirements practices. Addison-Wesley Longman Publishing Co., Inc., USA (2001)
31. Yu, E., Mylopoulos, J.: Understanding why in requirements engineering–with an example. In: Workshop on System Requirements: Analysis, Management, and Exploitation. pp. 4–7 (1994)
32. Yu, E.: Towards modeling and reasoning support for early-phase requirements engineering. Proceedings of the IEEE International Conference on Requirements Engineering pp. 226 – 235 (1997)