

DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

**A NEW APPROACH TO THE DEVELOPMENT
OF LICENSE MANAGEMENT SYSTEMS**

by
Hüseyin Tutku ÖZDİLEK

October, 2024

İZMİR

A NEW APPROACH TO THE DEVELOPMENT OF LICENSE MANAGEMENT SYSTEMS

**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for Master of Science in
Department of Computer Engineering, Computer Engineering Program**

**by
Hüseyin Tutku ÖZDİLEK**

October, 2024

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “**A NEW APPROACH TO THE DEVELOPMENT OF LICENSE MANAGEMENT SYSTEMS**” completed by **HÜSEYİN TUTKU ÖZDİLEK** under supervision of **ASSOC. PROF. DR. KÖKTEN ULAŞ BİRANT** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....
Assoc. Prof. Dr. Kökten Ulaş BİRANT

Supervisor

.....
Prof. Dr. Ayşegül ALAYBEYOĞLU

Jury Member

.....
Assoc. Prof. Dr. Özlem VARLIKLAR

Jury Member

Prof. Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENT

I would like to express my sincere appreciation to those who have contributed to the completion of this thesis.

I am deeply grateful to my mother, Özlem EKE, for her unwavering support and the foundation she has provided throughout my life. Her dedication has been instrumental in my academic journey.

I extend my heartfelt thanks to my sister Öykü ÖZDİLEK for her consistent support and understanding, which have been invaluable to me.

I wish to express my deep gratitude to my supervisor, Assoc. Prof. Kökten Ulaş BİRANT for his guidance and insightful advice throughout this research. His expertise and direction have contributed significantly to the quality of this work.

I also wish to acknowledge the committee members for their valuable comments and constructive feedback during my defense. Their input has significantly enhanced the outcome of this thesis.

Finally, I would like to thank Burak YENİER and İlker EKE for their advice and support, which have been crucial in balancing my professional and academic commitments.

Thank you all for your contributions to this endeavor.

Hüseyin Tutku ÖZDİLEK

A NEW APPROACH TO THE DEVELOPMENT OF LICENSE MANAGEMENT SYSTEMS

ABSTRACT

This thesis presents a comprehensive approach to addressing the complexities of software license management within modern organizations. Managing associated licenses has become a critical yet challenging task as software becomes increasingly integral to business operations. This thesis proposes the development of a License Management System (LMS) that integrates advanced technologies such as blockchain, smart cards, and autonomic computing to create a robust, scalable, and secure solution. The LMS is designed to streamline the processes of license selection, compliance, and management, addressing the evolving legal landscape and the diverse ecosystems in which software operates. Additionally, the system offers practical tools such as a license compatibility checker and a centralized database for managing license agreements and dependencies. The thesis also explores potential future enhancements, including integrating artificial intelligence for predictive analysis and automated compliance checks, as well as the expansion of blockchain-based license enforcement through smart contracts. These enhancements are intended to adapt to the dynamic nature of software development and the continuous evolution of licensing models. By simplifying license management and ensuring legal compliance, the proposed system aims to reduce administrative burdens and mitigate risks associated with non-compliance. Furthermore, the thesis discusses the broader implications of the LMS for the software industry, highlighting its potential to shape the future of software license management and foster a more secure and efficient digital landscape.

Keywords: Software license management, blockchain, smart cards, autonomic computing, compliance, artificial intelligence, decentralized systems, license enforcement.

LİSANS YÖNETİM SİSTEMLERİNİN GELİŞTİRİLMESİNE YENİ BİR YAKLAŞIM

ÖZ

Bu tez, modern organizasyonlar içinde yazılım lisans yönetiminin karmaşıklıklarını ele alan kapsamlı bir yaklaşım sunmaktadır. Yazılım, iş süreçlerinin ayrılmaz bir parçası haline geldikçe, ilgili lisansların yönetimi kritik ancak zorlu bir görev haline gelmiştir. Bu tez, blok zinciri, akıllı kartlar ve otonom bilişim gibi ileri teknolojileri entegre eden, sağlam, ölçeklenebilir ve güvenli bir Lisans Yönetim Sistemi (LMS) geliştirilmesini önermektedir. LMS, lisans seçimi, uyumluluk ve yönetim süreçlerini basitleştirerek, yazılımın faaliyet gösterdiği çeşitli ekosistemler ve gelişen yasal ortamlarla uyum sağlamayı hedeflemektedir. Ayrıca, sistem, lisans uyumluluğu denetleyicisi ve lisans anlaşmalarını ve bağımlılıklarını yönetmek için merkezi bir veri tabanı gibi pratik araçlar sunmaktadır. tez ayrıca, yapay zeka entegrasyonu ile tahmin analizi ve otomatik uyumluluk kontrolleri gibi gelecekteki olası geliştirmeleri ve akıllı sözleşmeler yoluyla blok zinciri tabanlı lisans uygulamalarının genişletilmesini araştırmaktadır. Bu geliştirmeler, yazılım geliştirme süreçlerinin dinamik doğasına ve lisans modellerinin sürekli evrimine uyum sağlamak amacıyla tasarlanmıştır. Önerilen sistem, lisans yönetimini basitleştirerek ve yasal uyumluluğu sağlayarak idari yükleri azaltmayı ve uyumsuzlukla ilgili riskleri en aza indirmeyi amaçlamaktadır. Ayrıca, tez, LMS'nin yazılım endüstrisi üzerindeki daha geniş etkilerini tartışarak, yazılım lisans yönetiminin geleceğini şekillendirme potansiyelini ve daha güvenli ve verimli bir dijital ortamı teşvik etme kabiliyetini vurgulamaktadır.

Anahtar Kelimeler: Lisans yazılımı yönetimi, blok zinciri, akıllı kartlar, otonom bilişim, uyumluluk, yapay zeka, merkeziyetsiz sistemler, lisans uygulama.

CONTENTS

	Page
THESIS EXAMINATION RESULT FORM.....	iii
ACKNOWLEDGMENT.....	iv
ABSTRACT.....	v
ÖZ.....	vi
CONTENTS.....	vii
LIST OF FIGURES.....	viii
 CHAPTER ONE - INTRODUCTION.....	 1
1.1 The Evolution of Software Licensing.....	1
1.2 The Importance of License Selection.....	2
1.3 Challenges in License Management.....	3
1.4 Need For An Integrated Solution.....	4
1.5 Scope and Objectives of the Project.....	5
1.6 Summary.....	5
 CHAPTER TWO - PROBLEM STATEMENT.....	 7
 CHAPTER THREE - PREVIOUS STUDIES.....	 10
3.1 An Ontological View.....	10
3.1.1 System Overview.....	11
3.1.2 License Types.....	11
3.1.3 License Agreement Representation.....	11
3.1.4 Task Ontology.....	12
3.1.5 Example Ontology.....	12
3.2 Software License Management In The Digital Era And Possible Risks In Enterprises.....	13
3.2.1 Methodology.....	13
3.2.2 Current Situation Analysis.....	14

3.2.3 Identified Risks.....	14
3.2.4 Risk Mitigation Strategies.....	15
3.2.5 Cost Benefit Analysis.....	15
3.3 Framework For Automated License Clearance.....	16
3.3.1 State Of The Art In Automated Rights Clearance.....	16
3.3.2 DALICC Software Architecture & Functional Spectrum.....	17
3.3.3 Data Modelling.....	17
3.3.4 Reasoning Over Licenses.....	18
3.3.5 Assessment of the DALICC Framework.....	18
3.3.6 Future of the DALICC Framework.....	18
3.4 Decentralized Approach to Software Licensing.....	19
3.4.1 System Model.....	19
3.4.2 License Contract Scheme.....	20
3.4.3 Implementation and Operations.....	20
3.4.4 Experimental Evaluation.....	21
3.4.5 Cost Analysis.....	21
3.4.6 Possible Enhancements.....	22
CHAPTER FOUR - PROPOSED SOLUTION.....	23
4.1 License Manager.....	25
4.1.1 Entities and Relationships.....	26
4.2 Product.....	29
4.2.1 Entities and Relationships.....	29
4.3 Software Vendor.....	31
4.3.1 Entities and Relationships.....	31
CHAPTER FIVE - PROJECT CODE.....	34
5.1 License Manager Models.....	34
5.2 Product Models.....	36

5.3 Software Vendor Models.....	38
CHAPTER SIX - COMPARISON.....	41
6.1 Core Objectives.....	41
6.2 Methodologies.....	42
6.3 Unique Contributions.....	42
6.4 Comparison of Effectiveness.....	43
CHAPTER SEVEN - CONCLUSION.....	45
7.1 Synthesis of Findings.....	45
7.2 Contributions and Implications.....	46
7.3 Summary.....	47
CHAPTER EIGHT - FUTURE ENHANCEMENTS.....	48
8.1 Integration of Advanced Automation and Artificial Intelligence.....	48
8.2 Expansion to Blockchain-Based License Enforcement.....	49
8.3 Enhanced User Experience With Smart Card Technology.....	49
8.4 Expansion of Licensing Models and International Compliance.....	50
8.5 Integration With Emerging Technologies.....	50
8.6 Summary.....	51
REFERENCES.....	52

LIST OF FIGURES

	Page
Figure 4.1 Flow Chart Diagram of the Project.....	24
Figure 4.2 License Manager ER-Diagram.....	26
Figure 4.3 Product ER-Diagram.....	30
Figure 4.4 Software Vendor ER-Diagram.....	32
Figure 5.1 License Manager Models.....	34
Figure 5.2 Product Models.....	36
Figure 5.3 Software Vendor Models.....	38

CHAPTER ONE

INTRODUCTION

In the contemporary digital era, software has become the backbone of innovation and operational efficiency across virtually every industry. As organizations increasingly rely on software to drive their business processes, the management of software licenses has emerged as a critical yet complex task. Software licenses not only define the legal framework within which software can be used, modified, and distributed but also significantly influence the evolution, integration, and reuse of software across diverse ecosystems. The importance of effective software license management cannot be overstated, as it plays a crucial role in ensuring compliance with legal obligations, optimizing software investments, and mitigating the risks associated with non-compliance.

1.1 The Evolution of Software Licensing

The landscape of software licensing has evolved significantly over the past few decades, driven by changes in legal frameworks, technological advancements, and shifts in software development practices. Initially, software licenses were relatively simple, focusing primarily on restricting unauthorized copying and distribution. However, as software became more complex and integral to business operations, licensing models also grew more sophisticated. The emergence of open-source software introduced new licensing paradigms, such as permissive licenses (e.g., MIT, Apache) and copyleft licenses (e.g., GPL), each with its own set of rules and implications for software usage and distribution.

As detailed in the study “An Exploratory Study of the Evolution of Software Licensing,” the evolution of software licensing has been shaped by several factors, including the growing importance of intellectual property rights, the need for interoperability among software systems, and the desire to foster innovation through collaborative development models (Di Penta et al., 2010).

This evolution has facilitated the growth and diversity of software but has also introduced new challenges in license selection, compliance, and management. Developers and organizations must now navigate a complex web of licensing options, each with its own legal and operational implications, making selecting and managing software licenses a pivotal yet challenging task.

1.2 The Importance of License Selection

Selecting a software license is one of the first and most critical decisions a developer or development team must make. This decision determines how others can use, modify, and distribute the software and can significantly influence the project's adoption and contribution rate. For example, permissive licenses like the MIT or Apache licenses generally allow for broader use and integration of the software, making them attractive for projects that aim to maximize adoption and encourage commercial use. In contrast, copyleft licenses like the GPL ensure that derivative works remain open source, promoting a collaborative development environment but potentially limiting commercial exploitation.

The choice between these licensing models is not merely a legal decision but also a strategic one that can have far-reaching implications for the software's future. As Vendome C. et al. (2015) highlighted in their study "When and Why Developers Adopt and Change Software Licenses," developers often adopt licenses early in the project's lifecycle. However, they may change these licenses as the project evolves. Various factors, including community influence, commercial reuse considerations, and compatibility with dependencies can drive this evolution. For instance, a project initially released under a permissive license might later switch to a copyleft license to ensure contributions remain open source, or vice versa, to enable broader commercial integration. This fluidity in licensing choices underscores the need for a dynamic and adaptable approach to license management.

1.3 Challenges in License Management

Managing software licenses is not a static task but an ongoing process that requires continuous attention and vigilance. As projects grow and evolve, so too do their dependencies and the licenses governing them. The inclusion of third-party libraries, each with its own licensing terms, can lead to conflicts if not managed carefully. For example, a permissive license may conflict with a more restrictive one, potentially leading to legal disputes or the loss of valuable intellectual property. Therefore, developers must not only understand the terms of their primary license but also ensure compatibility with the licenses of all incorporated dependencies. Effective license management is crucial in reducing both costs and risks associated with software usage in organizations, as highlighted by Gull and Wehrmann (2009), who propose a comprehensive model for combining different license types to optimize cost efficiency while maintaining service quality.

Moreover, the complexity of modern software ecosystems, characterized by the integration of multiple software components from different sources, exacerbates the challenges of license management. The findings from Durdu and Tutğaç's (2022) study on software license management in large enterprises reveal that inadequate license management can lead to significant financial and operational risks, including non-compliance penalties, legal disputes, and inefficient use of software assets. In particular, the study highlights the risks associated with inactive (idle) software licenses, duplicate procurement, and external audits, which can result in substantial financial losses if not properly managed.

Introducing new technologies, such as cloud computing and the Internet of Things (IoT), further complicates the license management landscape. Traditional license management practices may not be well-suited to these new environments, where software is often distributed across multiple platforms and accessed by users in different regions. Magnanini et al.(2019) noted that the decentralized nature of these technologies requires a more flexible and secure approach to license management, one that can adapt to the specific needs of these emerging platforms. For instance,

blockchain technology offers a promising solution by providing a secure, tamper-proof system for managing licenses. At the same time, smart cards can enhance security by offering a tamper-resistant method for storing and transferring licenses.

1.4 The Need For An Integrated Solution

With the growing need for adaptable software solutions, the implementation of autonomic software license management systems, as demonstrated by Noorian and Perry (2009), offers a robust approach to managing license patterns effectively within enterprise environments. Such a solution must be capable of handling the diverse licensing models and compliance requirements that organizations face today while also being adaptable to future changes in technology and regulatory environments. The proposed solution in this thesis is a comprehensive web application designed to address these needs by providing a centralized platform for managing all aspects of software licensing.

The proposed system will integrate advanced technologies, such as blockchain for secure license management, AI for automated compliance checks, and smart cards for secure license storage and transfer. These technologies will work together to create a dynamic and scalable system that can adapt to the changing needs of developers and organizations. For example, the system's AI component will be able to analyze licensing agreements in real time, flagging potential conflicts and suggesting optimal licensing strategies based on the specific needs of the project. Meanwhile, the blockchain component will ensure that all licensing transactions are securely recorded and easily auditable, reducing the risk of non-compliance and enhancing transparency.

In addition to its technical features, the proposed system will also offer a range of practical tools designed to simplify the day-to-day management of software licenses. These tools will include a license compatibility checker, a database of common licenses with detailed descriptions and use cases, and a set of tracking and

management tools for dependencies' licenses. By integrating these functionalities, the system aims to provide developers with a user-friendly interface that makes it easy to select, manage, and track software licenses throughout the software development lifecycle.

1.5 Scope and Objectives of the Project

This project aims to provide a detailed overview of the proposed software license management system, including its architecture, key features, and potential applications. The thesis will begin with a review of the existing literature on software license management, highlighting the current challenges and opportunities in the field. This thesis will be followed by a detailed description of the proposed system, including its technical components, design principles, and expected benefits.

In the subsequent sections, the thesis will explore the potential future enhancements to the system, such as the integration of AI for predictive analysis, the expansion of blockchain-based license enforcement, and the development of multi-function smart cards for license storage. These enhancements are designed to address the evolving needs of developers and organizations, ensuring that the system remains relevant and effective in the face of changing technological and regulatory landscapes.

Finally, the thesis will conclude with a discussion of the broader implications of the proposed system for software development and management. By providing a comprehensive and adaptable solution to software license management challenges, the proposed system can significantly enhance the efficiency, security, and compliance of software development processes, ultimately contributing to the growth and sustainability of the software industry.

1.6 Summary

The introduction has outlined the critical importance of effective software license management in today's complex and rapidly evolving software landscape. By

integrating advanced technologies and offering a comprehensive set of tools, the proposed system aims to provide a scalable, secure, and user-friendly solution to the challenges of software license management. The following sections will delve deeper into the specifics of the proposed system, exploring how it can be implemented and enhanced to meet the needs of developers and organizations both now and in the future.

CHAPTER TWO

PROBLEM STATEMENT

The selection and management of software licenses are pivotal yet complex activities within the software development lifecycle. These activities are not only foundational to the legal distribution and use of software but also influence software's evolution, integration, and reuse across diverse ecosystems. As detailed in "An Exploratory Study of the Evolution of Software Licensing," software licensing has evolved significantly, impacted by changes in legal landscapes, development practices, and community standards. This evolution, while facilitating the growth and diversity of software, introduces challenges in license selection, compliance, and management for developers.

Selecting a software license is one of the first critical decisions a developer or a development team makes. This decision determines how others can use, modify, and distribute the software. For instance, choosing between permissive licenses (like MIT or Apache) and copyleft licenses (like GPL) can significantly influence the project's adoption and contribution rate. Permissive licenses generally allow for broader use and integration, while copyleft licenses ensure that derivative works remain open source, promoting a collaborative development environment but potentially limiting commercial use.

Moreover, managing software licenses is not a static task. As projects grow and evolve, so too do their dependencies and the licenses governing them. The findings from "When and Why Developers Adopt and Change Software Licenses" by Vendome C. et al. (2015) reveal that developers often adopt licenses early in the project's lifecycle but may change these licenses as the project evolves. Various factors, including community influence, commercial reuse considerations, and compatibility with dependencies can drive this evolution. For example, a project initially released under a permissive license might later switch to a copyleft license to

ensure contributions remain open source, or vice versa, to enable broader commercial integration.

Additionally, compliance with chosen licenses requires ongoing attention. The inclusion of third-party libraries, each with its own licensing terms, necessitates careful management to avoid conflicts. Failure to comply with license terms can lead to legal disputes and can jeopardize the project's integrity and reputation. Therefore, developers must not only understand the terms of their primary license but also ensure compatibility with the licenses of all incorporated dependencies. As Zhao and Perry (2008) noted in their study on autonomic license management, the complexity of these licensing agreements can often overwhelm developers, leading to potential oversights and non-compliance risks.

Another layer of complexity arises from the various licensing models that software vendors use, such as time-based, concurrent, consumptive, and cumulative licenses. Each presents unique challenges in terms of compliance and management. For instance, time-based licenses require strict adherence to usage periods, while concurrent licenses limit the number of users accessing the software simultaneously. These varying models necessitate a robust management system that can handle the intricacies of each license type and ensure that all terms are met.

Moreover, as businesses increasingly adopt cloud-based services and other modern IT infrastructures, traditional license management practices are often inadequate. The transition to these new models requires a flexible and dynamic approach to license management that can adapt to the specific needs of different environments. Magnanini et al. (2019) discussed the use of decentralized technologies like blockchain, which offers a potential solution by providing a secure, tamper-proof system for managing licenses across various platforms and jurisdictions.

The introduction of smart cards as a means of securing software licenses adds yet another dimension to the problem. As Aura and Gollmann (1999) highlight, smart

cards can provide a secure and tamper-resistant way to store and transfer licenses. However, they also introduce challenges related to user accessibility and system integration. The need to balance security with usability is a recurring theme in license management, and any effective solution must address this tension.

In light of these complexities, this thesis proposes developing a web application to address these challenges. The goal is to provide a comprehensive solution that simplifies the process of license selection, facilitates the management of license compatibility issues, and supports informed decision-making regarding license changes throughout the software development lifecycle. This application would offer features such as a license compatibility checker, a database of common licenses with detailed descriptions and use cases, and tools for tracking and managing dependencies' licenses. The application aims to streamline license management by integrating these functionalities, ensuring legal compliance, and fostering a more robust and adaptable software development process. This approach not only addresses the immediate needs of developers but also lays the groundwork for more advanced license management practices that can evolve alongside the software industry.

In conclusion, the complexity of software license management, driven by the diversity of licensing models, the evolution of software ecosystems, and the integration of new technologies, requires a sophisticated and adaptable approach. The proposed solution in this thesis seeks to meet these demands, providing a tool that simplifies and automates key aspects of license management, thus enabling developers to focus more on innovation and less on the administrative burdens associated with licensing.

CHAPTER THREE

PREVIOUS STUDIES

3.1 An Ontological View

In the evolving landscape of digital license management, the efficiency and effectiveness of license agreements have become a focal point for both software vendors and customers. The paper "An Ontology for Autonomic License Management" by Qian Zhao and Mark Perry (2008) presents a pioneering approach to this challenge, aiming to bridge the gap between human-readable license agreements and their machine-readable representations through the development of an ontology.

Zhao and Perry's study is situated at the intersection of autonomic computing and license management. The primary goal is to automate the interpretation and enforcement of license agreements to reduce operational costs and mitigate risks of non-compliance. They identify the dual challenge of maintaining the readability of license agreements for humans while ensuring their machine readability and unambiguous interpretation by autonomic systems. The development of an ontology serves as the cornerstone of their proposed solution, facilitating a structured and extensible representation of license terms and conditions.

The authors introduce the concept of ontology as a formal knowledge representation of licensing. This approach not only meets the representation requirements but also offers improvements to knowledge reusability due to the inherent sharing nature of such representations. They employ XML-based ontology languages such as OWL (Web Ontology Language), which are user-friendly for non-developers often responsible for implementing and managing license agreements.

3.1.1 System Overview

Zhao and Perry's ontology is integrated into a license management system built on the SmArt (Semantic Agreement) framework. This framework leverages ontology to represent license agreements and other shared knowledge, employing agents to carry out tasks. The SmArt system is designed to operate efficiently, allowing data centers to manage licenses with minimal human intervention.

3.1.2 License Types

The study identifies five primary license types commonly used in data centers:

- *Time-based License*: Valid between specified start and end times.
- *Concurrent License*: A limited number of reusable licenses available at any given time.
- *Consumptive License*: A fixed number of non-reusable licenses available throughout the subscription period.
- *Cumulative License*: Non-reusable licenses with no limit during the subscription period, often used in pay-per-use models.
- *Named License*: Valid for a specific group of users, checked against a predefined list.

3.1.3 License Agreement Representation

The domain ontology in the SmArt system includes five basic concepts: Agreement, Service, Resource, Constraint, and Request. The license agreement is structured with a Head (Service, Customer, Vendor) and a Body containing a list of provision items (LAItem), which are detailed further using the Backus-Naur Form (BNF).

3.1.4 Task Ontology

The task ontology defines how the system operates, including four main subsystems:

- *Service Management*: Interfaces with customers, handling requests and dispatching tasks to other subsystems.
- *Service Subscription*: Manages subscription requests and updates the system knowledge core.
- *Service Configuration*: Verifies service requests against license agreements and configures the service accordingly.
- *Service Observation*: Monitors and controls the service usage, ensuring compliance with license agreements.

3.1.5 Example Ontology

An example scenario illustrates the practical application of ontology. A company provides a service with specific license terms, such as a time-based license allowing 500 concurrent users for a year. The ontology defines tuples representing these terms, which the system uses to manage the license dynamically.

The ontology-based approach presented by Zhao and Perry offers a flexible and extensible solution for autonomic license management. The use of OWL/RDF enhances the system's ability to accommodate new license types and adapt to changing requirements without significant modifications. Future work involves extending this framework to manage other resources and constraints beyond the simplified prototypes discussed in the study.

By automating the interpretation and enforcement of license agreements, this ontology-based system reduces operational costs and mitigates compliance risks, providing a robust solution for modern data centers and software vendors.

3.2 Software License Management In The Digital Era And Possible Risks In Enterprises

Durdu and Tutğaç focused on identifying the risks associated with software licenses in large enterprises and proposing strategies to mitigate these risks. Their study aimed to highlight the importance of effective risk management strategies in software license management.

Their approach included a thorough examination of common risks faced by enterprises in managing software licenses, such as financial losses due to non-compliance, duplicate procurements, and idle software. The study proposed the establishment of a dedicated Software Asset Management (SAM) team and the implementation of a comprehensive SAM system as key strategies for risk mitigation.

In the context of digital transformation, enterprises are making significant investments in software assets, making the management of these assets a critical operational and strategic imperative. Software licenses, with their varied and complex models, represent a significant portion of these investments. The dynamic nature of software licensing, coupled with the strategic shift towards enterprise risk management, necessitates a more integrated approach to effectively managing these assets and their associated risks.

3.2.1 Methodology

Durdu and Tutğaç's study was applied to a large-scale enterprise in the telecommunications sector, which provided a comprehensive overview of the current state of software asset management within the company. The enterprise, with over 30000 employees and serving more than 40 million customers, offered a substantial case study for examining the complexities and risks associated with software licenses.

3.2.2 Current Situation Analysis

The study employed two methods to identify risks in IT software license management: one-on-one interviews and focus group meetings. A detailed inventory of software licenses was created by contacting 166 units within the company and administering a set of 25 questions about the software, license metrics, validity periods, and other relevant details. The analysis revealed that the company had over 1600 different types of software licenses from more than 100 vendors, with 132 different license models.

3.2.3 Identified Risks

The focus group meetings, attended by five experts in contract and license management, identified twelve significant risks associated with software license management. These risks were categorized and scored based on their probability and potential impact. The key risks included:

- *Inactive (Idle) Software Cost Risks:* Financial losses due to unused software.
- *Duplicate Procurement Risks:* Financial losses from purchasing duplicate licenses.
- *External Audit Risks:* Reputational and financial risks from software supplier audits.
- *Failure to Be Up to Date with Versions:* Security and business continuity risks are due to outdated software.
- *Ineffective Inventory Control:* Workforce loss from redundant inventory work.
- *Deficiency in Alternative Product Analysis:* Financial losses from not using cost-effective alternatives.
- *Long Procurement Processes:* Loss of workforce and bargaining power due to prolonged procurement processes.
- *Inadequate Contract Control:* Financial and business losses from poorly managed contracts.

- *Ineffective Contract Tracking:* Business continuity risks from poorly tracked contract periods.
- *Internal Audit Risks:* Workforce loss during internal audits due to inefficient software inventory management.
- *Inefficiency in Following the License Model:* Financial losses from delays adapting to new licensing models.
- *Risks from Incomplete Feasibility:* Financial and business losses from improperly analyzed software requirements.

3.2.4 Risk Mitigation Strategies

To address the identified risks, Durdu and Tutğaç proposed several strategies:

- *Establishment of a SAM Team:* Form a team dedicated to managing the software license inventory and ensuring it is up-to-date and accurate.
- *SAM System Setup:* Implementing a Software Asset Management Tool to manage licenses with actual data, including usage and installation data.
- *Software Life Cycle Process:* Defining and implementing a software life cycle process to clearly outline roles and responsibilities in software license management.

3.2.5 Cost-Benefit Analysis

The study conducted a cost-benefit analysis of implementing the proposed strategies. The initial investment for establishing a SAM system and team was estimated to be around 2 to 3 million TL, with an annual operational cost of 0.75 million TL. Potential savings justified this investment, as companies with SAM systems could save up to 30% in license expenditures, according to Gartner's analysis.

The study by Durdu and Tutğaç provides a comprehensive framework for managing software license risks in large enterprises. Companies can effectively mitigate risks and optimize their software license expenditures by adopting an

enterprise risk management approach and implementing dedicated SAM teams and systems. This approach not only reduces financial losses but also enhances business continuity and compliance with licensing agreements. The findings and recommendations from this study offer valuable insights for enterprises looking to improve their software asset management practices.

3.3 Framework For Automated License Clearance

The DALICC (Data Licenses Clearance Center) framework, developed by Tassilo Pellegrini and colleagues, represents a significant advancement in automated license clearance and management of digital assets. It supports legal experts, innovation managers, and developers in the legally secure reuse of third-party data sources. By enabling automated rights clearance, DALICC helps detect licensing conflicts and significantly reduces the costs associated with rights clearance in creating derivative works. Key components of the DALICC framework include a comprehensive license library, search functionality, a license composer for custom licenses, and a license substitutability check to ensure compatibility and compliance (Pellegrini et al., 2019).

The DALICC framework was designed to address the complexities of managing digital licenses, especially as IT applications increasingly utilize data from various sources. Effective license clearance is crucial to ensure legal compatibility and to facilitate the reuse of digital assets for both commercial and non-commercial purposes. The framework aims to streamline the resolution of licensing conflicts and reduce the transaction costs associated with manual clearance processes.

3.3.1 State Of The Art In Automated Rights Clearance

Licenses are deontic statements that govern the permissions, prohibitions, and obligations associated with digital assets. These licenses control the access to, usage of, and transactions involving digital assets. The spectrum of available licensing models ranges from property rights (all rights reserved) to public domain (no rights reserved).

Rights Expression Languages (RELs) play a vital role in managing digital rights. Notable REL vocabularies include MPEG-21, the Creative Commons Rights Expression Language (ccREL), and the Open Digital Rights Language (ODRL). Previous research efforts have focused on developing ontologies and rule-based engines to reason over licenses and detect conflicts. However, these approaches often lack comprehensive implementation and practical application.

3.3.2 Dalicc Software Architecture & Functional Spectrum

The DALICC framework comprises four main functional components:

- *License Library*: A repository containing both machine-readable (ODRL policies) and human-readable representations of licenses. This library supports SPARQL queries to present license properties to users in an accessible format.
- *License Search*: Allows users to search for licenses either by full-text or by selecting specific actions (permissions, duties, or obligations). This functionality leverages SPARQL queries and a reasoner to identify relevant licenses.
- *License Composer*: This enables users to create customized licenses by defining permissions, duties, and prohibitions through a set of questions. The resulting custom licenses can be downloaded as RDF files or text files.
- *License Substitutability Check*: Ensures that each license is compatible with the target license for derivative works. This check identifies conflicts at the action level of each license.

3.3.3 Data Modelling

The modeling process for DALICC involves four phases:

- *Analyzing the License Text*: Review the legal text of each license to identify permissions, duties, and prohibitions.

- *Defining Vocabularies*: Using ODRL and other vocabularies to express the license terms in a machine-readable format.
- *Deriving Modelling and Mapping Mechanisms*: Creating additional terms and mapping mechanisms to represent licenses accurately.
- *Defining a Common Model*: Establishing a unified model for annotating and comparing licenses using a dependency graph.

3.3.4 Reasoning Over Licenses

DALICC uses Answer Set Programming (ASP) to reason over licenses. The ASP system CLINGO checks the logical coherence of licenses, identifies equivalent and compatible licenses, and detects conflicts. The reasoner processes user queries and provides suitable license options by evaluating permissions, prohibitions, and duties.

3.3.5 Assessment Of The Dalicc Framework

DALICC underwent a pre-test involving 12 users (six software developers and six legal professionals) to ensure legal validity and usability. The feedback highlighted the complexity of the composer interface and the visualization of compatibility logs. These insights will guide further enhancements of the framework.

3.3.6 Future Of The Dalicc Framework

The DALICC framework offers a robust solution for automated license creation, compatibility checking, and substitutability assessment. Future work includes expanding the license library, utilizing reasoning capabilities for conflict resolution, enabling organizations to create custom applications using DALICC APIs, visualizing data workflows, and implementing security features using blockchain protocols.

The Austrian Research Promotion Agency funds the DALICC project (No. 855396) and aims to provide a comprehensive solution for managing digital licenses effectively.

3.4 Decentralized Approach To Software Licensing

Federico Magnanini, Luca Ferretti, and Michele Colajanni propose a groundbreaking approach to decentralized software licensing utilizing blockchain technology. Their system leverages smart contracts on a permissionless blockchain, such as Ethereum, to manage license sales, usage reporting, and payments in an automated, flexible, and secure manner. This approach eliminates the need for centralized control, reducing the risks of failure and enabling more efficient cross-border payments.

In a fully interconnected world where network-related services heavily rely on software, the management of software license agreements is critical for both software vendors and communication providers. Traditional methods of managing software licenses involve declarative licenses and periodic audits, which are expensive and rigid. The authors propose a novel decentralized approach using blockchain and smart contracts, which promises a secure and cost-effective solution without a central point of failure. This system is designed to regulate relationships among untrusted parties efficiently.

3.4.1 System Model

The system primarily focuses on modern Communication Service Providers (CSPs) and Network Software Vendors (NSVs). CSPs provide network connectivity using virtualized network appliances, while NSVs sell software related to network services. The proposed system aims to handle software licensing in these network environments by leveraging the decentralization and security features of blockchain technology.

Key Requirements:

- *Usage Reporting:* The NSV requires resource usage reports from the CSP to assess compliance with the license agreement.
- *Service Continuity:* The CSP needs to ensure continuous service availability, which necessitates a highly available and reliable licensing infrastructure.

The proposed system uses the Ethereum blockchain to manage the entire life cycle of licenses, record authenticated usage information, and enable trustless license payments.

3.4.2 License Contract Scheme

The system architecture consists of license contracts on the Ethereum blockchain. Each license contract includes:

- CSP address
- License ID
- Price in ethers
- Maximum usage threshold
- Current usage value
- Status (compliant or non-compliant)

The license contract supports the following main operations:

- *Purchase*: CSPs purchase licenses by paying ethers to the license contract.
- *UpdateMetric*: CSPs update the current usage value of the license.
- *UpdateTerms*: NSVs update the terms of use of the license contract.
- *GetLicenseInfo*: A public read-only method that returns information about the license.

3.4.3 Implementation And Operations

The Purchase operation verifies the payment amount, generates a new license ID, and stores it. The UpdateMetric method updates the usage value and checks compliance with the maximum usage threshold. The implementation ensures minimal operational costs by storing only essential information on the blockchain.

3.4.4 Experimental Evaluation

The system's performance was evaluated using a prototype implementation based on Ethereum. The evaluation considered commit latencies influenced by transaction gas prices. Three categories of gas prices were used: fast, standard, and low, with corresponding commit latencies and operational costs.

3.4.5 Cost Analysis

The costs of the license contract operations were calculated based on gas prices at the time of the evaluation. The results showed that the most expensive operation is the creation of the license contract, followed by the Purchase operation. Bulk update strategies were proposed to reduce operational costs by updating usage metrics at regular intervals.

The evaluation considered a scenario with a private cloud provider hosting 90,000 VMs on 8,000 servers. The proposed system's costs were analyzed based on two approaches: individual updates for each VM event and bulk updates at regular intervals. The bulk update approach significantly reduced operational costs and was deemed viable for managing license network software services and virtualized functions.

The proposed system offers a secure and decentralized platform for managing software licensing in high-availability and flexibility scenarios. It leverages the Ethereum blockchain to handle payments and resource usage accounting in pay-as-you-grow licensing models. The prototype implementation demonstrated the feasibility and cost-effectiveness of the approach. Future work includes extending the system to handle license enforcement and integrating blockchain-specific solutions for enhanced security.

3.4.6 Possible Enhancements

- *Extending the License Library:* Including additional standard licenses.
- *Conflict Resolution:* Utilizing reasoning components for conflict resolution.
- *API Integration:* Providing organizations with APIs to create custom applications.
- *Visualizing Data Workflows:* Taking into account license provenance information.
- *Security Features:* Implementing blockchain protocols for secure license management.

In summary, Magnanini, Ferretti, and Colajanni's approach presents a novel and efficient method for managing software licenses using blockchain technology. It addresses critical challenges in the software licensing domain while offering significant benefits in terms of security, flexibility, and cost-efficiency.

CHAPTER FOUR

PROPOSED SOLUTION

In the contemporary digital landscape, the efficient management of software licenses, products, and their associated versions has become an essential aspect of organizational operations. As businesses increasingly rely on a diverse array of software solutions to support their core functions, the complexities of managing these digital assets have grown exponentially. This thesis presents a comprehensive solution designed to address the challenges associated with software license management, product cataloging, and version control within an organizational framework.

The proposed system aims to streamline the processes of managing software licenses and products by introducing a structured and systematic approach. This includes tracking software vendors, administering licenses, and managing the lifecycle of software products. The system's architecture is built around key entities such as Software Vendors, Products, Licenses, and License Assignments, each playing a crucial role in ensuring that software usage within an organization is both compliant and efficient.

One of the core components of this solution is the License Manager, which is responsible for overseeing the management of software licenses, including the assignment of licenses to users or entities, and ensuring that licenses are valid and up-to-date. The system also includes comprehensive product management capabilities, allowing organizations to catalog products, manage product versions, and control access rights to sensitive product information.

By implementing this solution, organizations can achieve greater control over their software assets, reduce the risks associated with non-compliance, and ensure that their digital resources are utilized effectively. The following sections of this

thesis will delve into the detailed design and functionalities of the proposed system, outlining the entities, relationships, and models that underpin its operation.

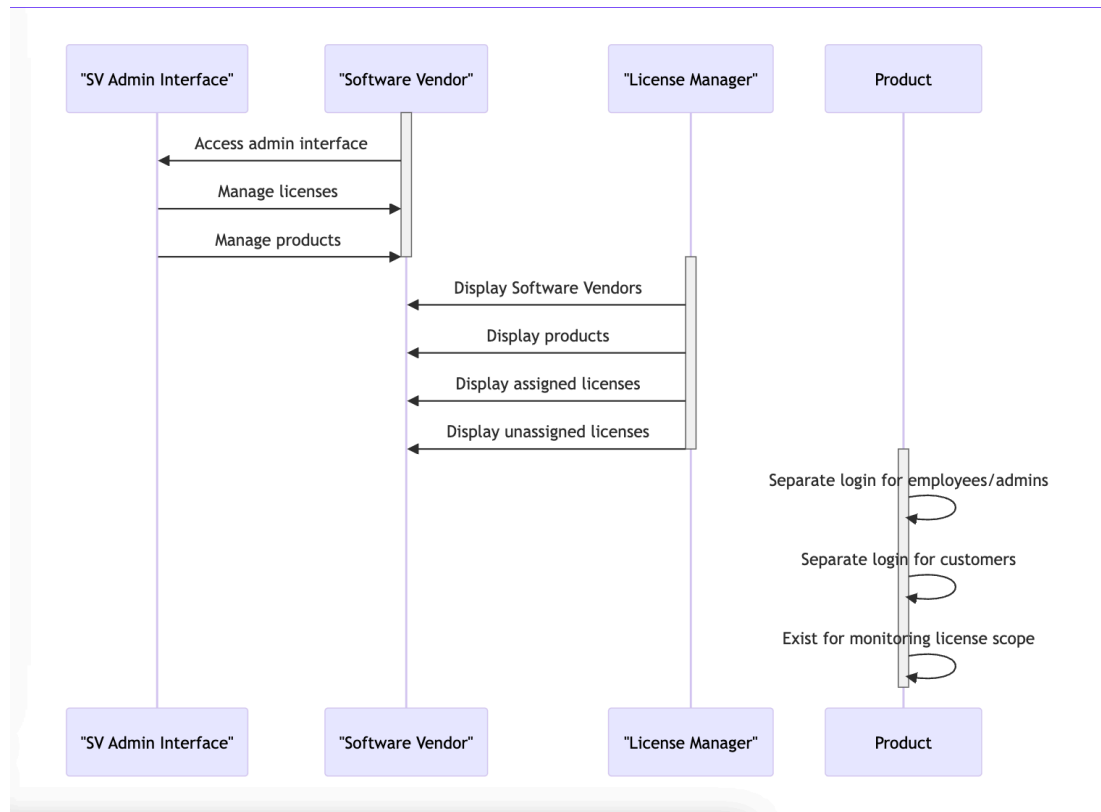


Figure 4.1 Flow Chart Diagram of the Project

The diagram illustrates the interaction between various components of a software licensing system through a sequence of actions:

- *SV Admin Interface to Software Vendor:* The SV Admin Interface, representing the administrative control for Software Vendors, begins by allowing access to the admin interface. Once logged in, the administrator can manage licenses and products associated with the software.
- *Software Vendor to License Manager:* The Software Vendor then interacts with the License Manager by sending requests to display the available software vendors. This process continues with the display of products, followed by displaying both assigned and unassigned licenses. These actions

likely help the License Manager to organize and manage licenses effectively, assigning them to appropriate users or entities as necessary.

- *License Manager to Product:* The License Manager component interacts with the Product component, which includes features for login functionality. The diagram distinguishes between separate logins for employees/admins and customers, ensuring that different user types have appropriate access levels. The product component is also responsible for monitoring the scope of the licenses and ensuring that the licenses are being used within their allowed limits.

Each interaction is represented as an arrow, indicating the flow of requests and responses between these components. The separation of login systems and license monitoring suggests a focus on secure, controlled access and compliance with licensing terms.

4.1 License Manager

The License Manager system is designed to manage software licenses in an organized and systematic way. The main objectives of this system include:

- *Managing Software Vendors:* The system tracks different software vendors, their products, and their registration status.
- *Tracking Products:* It maintains a detailed record of various software products provided by these vendors, including versioning and licensing status.
- *License Administration:* The system handles the creation and management of licenses for each product, including details such as license type and validity period.
- *Assigning Licenses:* It allows for assigning licenses to specific users or entities, storing detailed information about each assignment.

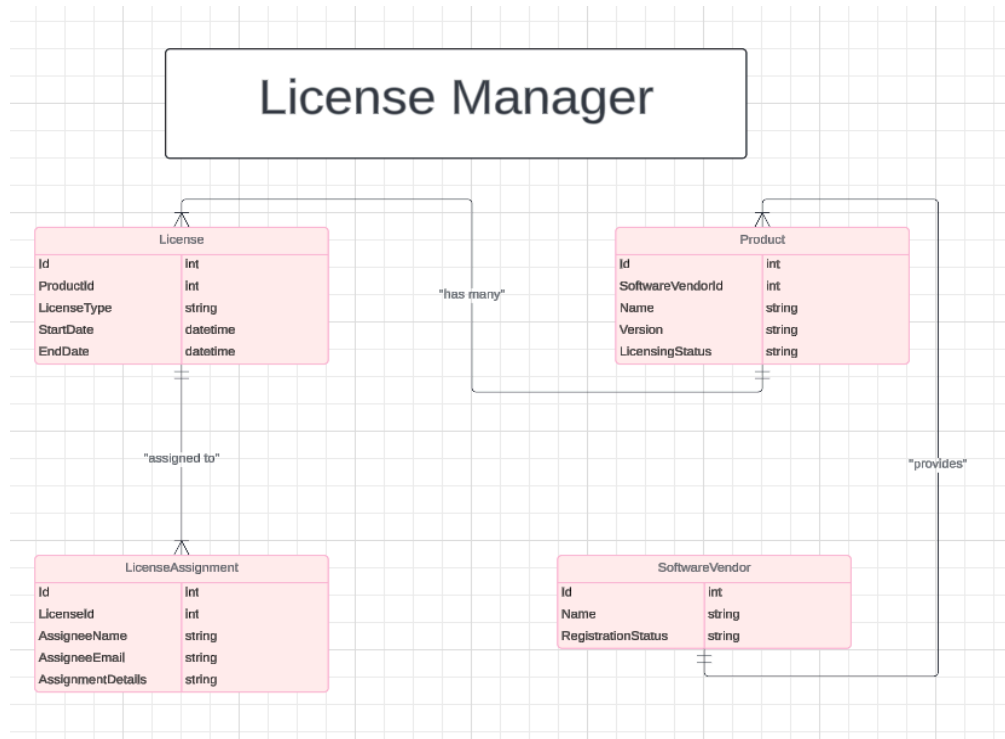


Figure 4.2 License Manager ER-Diagram

4.1.1 Entities and Relationships

a. Entities

Software Vendor

Purpose: The SoftwareVendor entity represents the companies or individuals who provide software products. This entity helps manage information about the sources of the software, ensuring that all products are properly attributed to their creators or distributors.

Key Functions:

- *Identification:* Each vendor is uniquely identified, allowing easy tracking and reference.
- *Registration Management:* The RegistrationStatus attribute helps monitor the vendor's status, which can be essential for ensuring that only authorized and compliant vendors are used.

- *Vendor-Specific Reporting:* By associating products and licenses with specific vendors, the system can generate vendor-specific reports, aiding in decision-making and compliance tracking.

Product

Purpose: The Product entity contains detailed information about the software products offered by the vendors. This entity is crucial for managing the catalog of available software and their specific versions and statuses.

Key Functions:

- *Catalog Management:* Tracks all products provided by vendors, ensuring a comprehensive inventory of available software.
- *Version Control:* The Version attribute helps manage different versions of the same product, which is critical for ensuring compatibility and support.
- *Licensing Information:* The LicensingStatus attribute provides a quick overview of whether a product is actively licensed, deprecated, or in some other state, aiding in the management of license renewals and expirations.

License

Purpose: The License entity represents the individual licenses issued for each product. This entity is central to the management of software usage rights, ensuring that the use of products is in compliance with licensing agreements.

Key Functions:

- *Usage Tracking:* Records the specific type of license (e.g., single-user, multi-user), which is essential for compliance and audit purposes.
- *Validity Period:* The StartDate and EndDate attributes define the active period of the license, helping manage renewals and ensuring that licenses are up-to-date.

- *Association with Products:* Each license is linked to a product, allowing for detailed tracking of which licenses apply to which software products.

License Assignment

Purpose: The LicenseAssignment entity keeps track of which licenses are assigned to which users or entities. This entity is critical for ensuring that licenses are properly allocated and that the system knows who is authorized to use each piece of software.

Key Functions:

- *Assignment Tracking:* By linking licenses to assignees, the system can ensure that all licenses are properly allocated and used by the intended individuals or groups.
- *Contact Information:* The AssigneeEmail attribute ensures that there is a way to contact the license holder, which can be important for notifications and compliance purposes.
- *Detailed Assignment Records:* The AssignmentDetails attribute allows for additional information about the assignment to be recorded, providing context and additional details that might be necessary for audits or troubleshooting.

b. Relationships

SoftwareVendor to Product: A single software vendor can provide multiple products. This relationship helps in organizing products under their respective vendors, facilitating better vendor management and reporting.

Product to License: Each product can have multiple licenses associated with it. This relationship ensures that each instance of product usage is tracked and managed according to its licensing terms.

License to LicenseAssignment: Each license can be assigned to multiple entities or individuals. This relationship is essential for tracking the allocation and usage of each license, ensuring compliance and proper management.

4.2 Product

The primary purpose of the system is to provide a structured framework for managing products, their associated content, and access rights within an organization. This model is essential for systems where products need to be cataloged and detailed with static content while also ensuring that access to product data is controlled and secure.

4.2.1 Entities And Relationships

a. Entities

Products

- *Cataloging Products:* The system can store and manage a list of products, each with unique identifiers, names, descriptions, and prices. This ensures that each product is distinctly identifiable and detailed.
- *Product Information Storage:* Detailed information about each product, such as its description and price, is maintained, facilitating comprehensive product data management.

Static Contents

- *Content Association:* Each product can have multiple static content entries associated with it. This allows for rich, detailed content to be linked to products, enhancing the user experience with additional information such as specifications, usage instructions, marketing content, and more.

- *HTML Content Storage:* Static content is stored in HTML format, making it suitable for web applications where rich text and multimedia content can be displayed effectively.

Access Control

- *Defining Access Levels:* The system can define different levels of access rights for each product. This means administrators can set who can view, edit, or manage specific product information.
- *Security Management:* By managing access rights, the system ensures that sensitive product information is protected and only accessible to authorized users. This is crucial for maintaining data integrity and security, especially in environments with multiple users or roles.

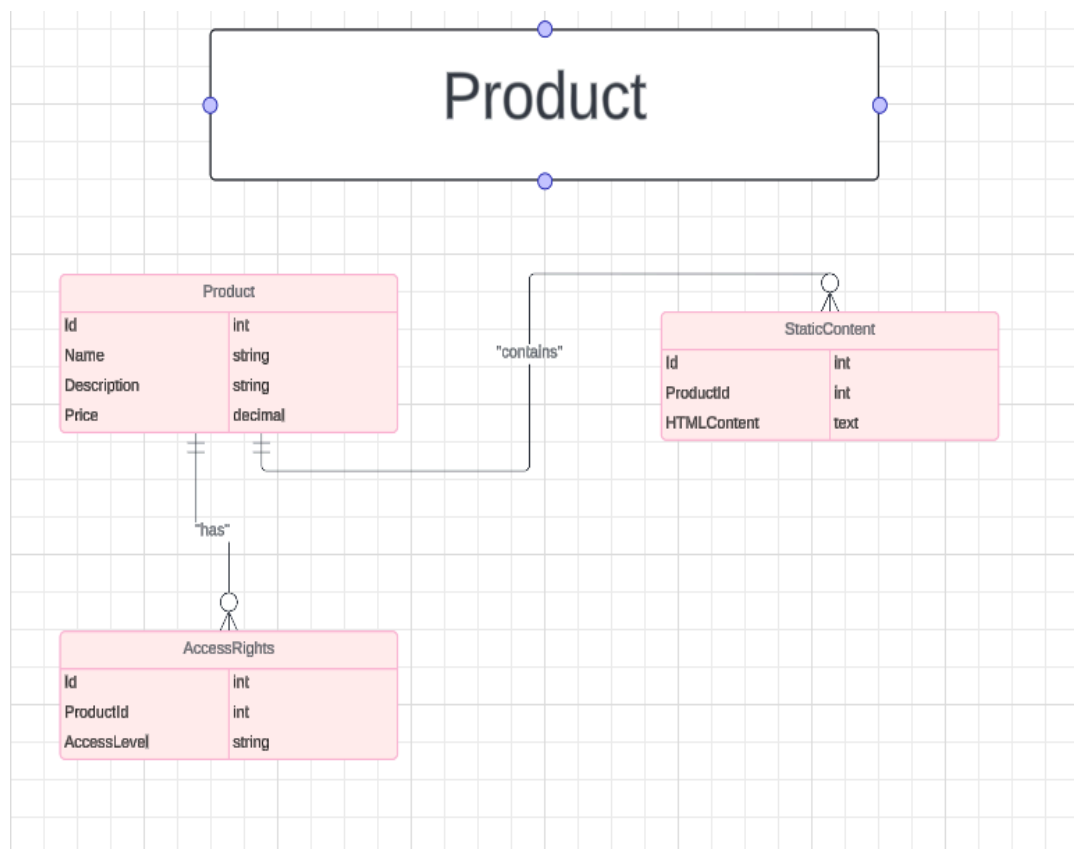


Figure 4.3: Product ER-Diagram

b. Relationships

Product depicts a system where products are the core entity. Each product can be associated with various static contents that enhance the information available to users. These contents are stored in HTML format, making them suitable for web-based applications where rich text and multimedia content are required.

Additionally, the model incorporates an access control mechanism through the AccessRights entity. This entity allows the system to define who can access and modify the product information, ensuring that sensitive or critical product data is protected and only accessible by authorized personnel.

4.3 Software Vendor

Software Vendor System where the primary focus is on managing software products and their versions. The functionality provided by this model includes, Product Management and Version Management.

4.3.1 Entities And Relationships

a. Entities

Products

Cataloging Products: The system can store and manage a list of software products, each with unique identifiers, names, development statuses, and access rights.

Tracking Development Status: The DevelopmentStatus attribute helps in tracking the current stage of each product, which is crucial for development and release planning.

Access By User

- *Access Control:* The system has the ability to control and manage which access rights or permissions a Software Vendor has over specific products.

Version

- *Versioning*: Each product can have multiple versions tracked over time. This is essential for software lifecycle management, where different versions may be released, maintained, or deprecated.
- *Release Information*: The system records critical information about each version, including its release date and locations of important files like the zipped package and reports. This ensures that all necessary details for each version are accessible and well-documented.

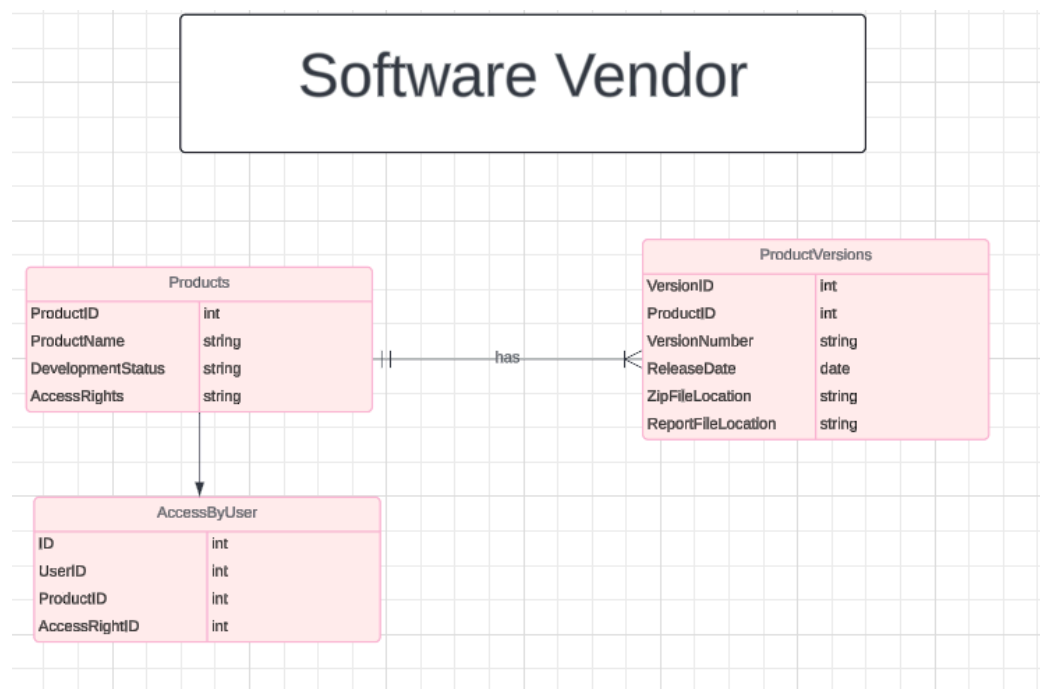


Figure 4.4 Software Vendor ER-Diagram

b. Relationships

This one-to-many relationship indicates that a single product can have multiple versions. Each version is associated with one product, but a product can have many versions.

The provided ER diagram offers a comprehensive data model for managing software products and their versions in a Software Vendor System. It supports detailed product cataloging, version tracking, and access control, making it an essential tool for software development and management.

This model ensures that all aspects of software products, from development status to version-specific details, are meticulously recorded and easily accessible, thereby enhancing the efficiency and effectiveness of software product management.

CHAPTER FIVE

PROJECT CODE

5.1 License Manager Models

```
class SoftwareVendor(models.Model):
    User = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    RegistrationStatus = models.CharField(max_length=50)
    ContactInfo = models.EmailField()

    def __str__(self):
        return self.User.username

class License(models.Model):
    LicenseType = models.CharField(max_length=50)
    IssueDate = models.DateField()
    ExpiryDate = models.DateField()
    LicenseCode = models.CharField(max_length=100)
    Status = models.CharField(max_length=50)
    Terms = models.TextField(null=True, blank=True)
    Product = models.ForeignKey(Product, on_delete=models.CASCADE)

    def __str__(self):
        return f"{self.LicenseType} - {self.Product.ProductName}"

class LicenseAssignment(models.Model):
    License = models.ForeignKey(License, on_delete=models.CASCADE)
    AssignedTo = models.CharField(max_length=100)
    Details = models.TextField(max_length=1000)

    def __str__(self):
        return f"{self.License} assigned to {self.AssignedTo}"
```

Figure 5.1: License Manager Models

Software Vendor

- *User*: A foreign key to the CustomUser model. This creates a many-to-one relationship, meaning each SoftwareVendor is associated with one CustomUser.
- *RegistrationStatus*: A character field with a maximum length of 50 characters, storing the registration status of the vendor.

- *ContactInfo*: An email field to store the contact information of the vendor.

License

- *LicenseType*: A character field with a maximum length of 50 characters, storing the type of the license.
- *IssueDate*: A date field to store the issue date of the license.
- *ExpiryDate*: A date field to store the expiry date of the license.
- *LicenseCode*: A character field with a maximum length of 100 characters, storing the unique code of the license.
- *Status*: A character field with a maximum length of 50 characters, storing the status of the license.
- *Terms*: A text field to store the terms and conditions of the license. It can be null or blank.
- *Product*: A foreign key to the Product model, creating a many-to-one relationship. Each license is associated with one product.

License Assignment

- *License*: A foreign key to the License model, creating a many-to-one relationship. Each assignment is associated with one license.
- *AssignedTo*: A character field with a maximum length of 100 characters, storing the entity (user or product) to which the license is assigned.
- *Details*: A text field with a maximum length of 1000 characters, storing additional details about the assignment.

Summary

The SoftwareVendor model represents vendors with a user, registration status, and contact information.

The License model represents licenses with various attributes, such as type, issue date, expiry date, code, status, terms, and associated product.

The LicenseAssignment model represents the assignment of licenses to users or products, with details about the assignment.

Each model includes a `__str__` method to provide a human-readable representation of the object, which is useful for debugging and displaying in the Django admin interface.

5.2 Product Models

```
class Product(models.Model):
    ProductName = models.CharField(max_length=100)
    Description = models.TextField()
    LicensingStatus = models.CharField(max_length=50)
    DevelopmentStatus = models.CharField(max_length=50)
    PortNumber = models.IntegerField(null=True, blank=True)
    AccessURL = models.URLField(null=True, blank=True)
    ContentType = models.CharField(max_length=50, null=True,
    blank=True)

    def __str__(self):
        return self.ProductName

    def get_producer(self):
        from LicenseManager.models import SoftwareVendor
        return SoftwareVendor.objects.get(id=self.Producer_id)

class AccessRight(models.Model):
    AccessLevel = models.CharField(max_length=50)

    def __str__(self):
        return self.AccessLevel

class StaticContent(models.Model):
    Product = models.ForeignKey(Product, on_delete=models.CASCADE)
    HTMLContent = models.TextField()

    def __str__(self):
        return f"Content for {self.Product.ProductName}"
```

Figure 5.2: Product Models

Product

- *ProductName*: A character field with a maximum length of 100 characters, storing the name of the product.
- *Description*: A text field to store a detailed description of the product.
- *LicensingStatus*: A character field with a maximum length of 50 characters, storing the licensing status of the product.
- *DevelopmentStatus*: A character field with a maximum length of 50 characters, storing the development status of the product.
- *PortNumber*: An integer field that can be null or blank, storing the port number associated with the product.
- *AccessURL*: A URL field that can be null or blank, storing the access URL for the product.
- *ContentType*: A character field with a maximum length of 50 characters that can be null or blank, storing the content type of the product.
- *get_producer*: A method to get the producer of the product. It lazily imports the SoftwareVendor model from the LicenseManager app and retrieves the producer using the Producer_id.

Access Right

- *AccessLevel*: A character field with a maximum length of 50 characters, storing the access level (e.g., Read, Write, Admin).

Static Content

- *Product*: A foreign key to the Product model, creating a many-to-one relationship. Each StaticContent is associated with one Product.
- *HTMLContent*: A text field to store static HTML content associated with the product.

Summary

The Product model represents products with various attributes such as name, description, licensing status, development status, port number, access URL, and content type. It includes methods to return the product name and to get the producer of the product.

The AccessRight model represents access rights with an access level attribute and includes a method for returning the access level.

The StaticContent model represents static HTML content associated with a product. It includes a method to return a string indicating the content is for the associated product.

5.3 Software Vendor Models

```
class ProductVersion(models.Model):
    Product = models.ForeignKey(Product, on_delete=models.CASCADE)
    VersionNumber = models.CharField(max_length=20)
    ReleaseDate = models.DateField()
    ZipFileLocation = models.FileField(upload_to='uploads/')
    ReportFileLocation = models.FileField(upload_to='reports/')
    def __str__(self):
        return f"{self.Product.ProductName} v{self.VersionNumber}"

class AccessByUser(models.Model):
    User = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    Product = models.ForeignKey(Product, on_delete=models.CASCADE)
    AccessRight = models.ForeignKey(AccessRight,
on_delete=models.CASCADE)

    def __str__(self):
        return f"{self.User.username} - {self.Product.ProductName} - {self.AccessRight.AccessLevel}"
```

Figure 5.3 Software Vendor Models

Product Version

- *Product*: A foreign key to the Product model, creating a many-to-one relationship. Each ProductVersion is associated with one Product.
- *VersionNumber*: A character field with a maximum length of 20 characters, storing the version number of the product.
- *ReleaseDate*: A date field to store the release date of the product version.
- *ZipFileLocation*: A file field to store the location of the zip file associated with the product version. The file is uploaded to the uploads/ directory.
- *ReportFileLocation*: A file field to store the location of the report file associated with the product version. The file is uploaded to the reports/ directory.

Access By User

- *User*: A foreign key to the CustomUser model, creating a many-to-one relationship. Each AccessByUser is associated with one CustomUser.
- *Product*: A foreign key to the Product model, creating a many-to-one relationship. Each AccessByUser is associated with one Product.
- *AccessRight*: A foreign key to the AccessRight model, creating a many-to-one relationship. Each AccessByUser is associated with one AccessRight.

Summary

The ProductVersion model represents different versions of a product with attributes like product, version number, release date, zip file location, and report file location. It includes a method to return a string combining the product name and version number.

The `AccessByUser` model represents the access rights of users to products with attributes like user, product, and access right. It includes a method to return a string combining the username, product name, and access level.

CHAPTER SIX

COMPARISON

To conduct a comparison between the projects discussed in the Literature Review section and the proposed solution, focusing on the core objectives, methodologies, and unique contributions of each can be a correct approach.

6.1 Core Objectives

“An Ontology for Autonomic License Management” by Zhao and Perry (2008): The primary objective is to create a system that automates the interpretation and enforcement of software licenses through an ontology-based approach. This system is designed to reduce operational costs and mitigate compliance risks by representing license agreements in a machine-readable format.

“Examination of Risks in Software License Management” by Durdu and Tutğaç (2022): This study aims to identify risks in managing software licenses in large enterprises and propose strategies for risk mitigation, focusing on implementing a Software Asset Management (SAM) system.

“DALICC Framework by Pellegrini et al. (2019)”: The objective is to automate the clearance of digital licenses to reduce costs and ensure legal compliance, focusing on creating a comprehensive license library and automated conflict detection.

“Decentralized Software Licensing by Magnanini et al. (2019)”: The goal is to utilize blockchain technology to manage software licenses in a decentralized manner, ensuring secure, flexible, and automated license management without centralized control.

Proposed Solution: The License Manager system aims to manage software licenses systematically, focusing on tracking software vendors, products, license administration, and assignment. The solution seeks to streamline the process of managing licenses in an organized and efficient manner, addressing issues like license assignment, compliance, and vendor-specific reporting.

6.2 Methodologies

Ontology-Based Approach: Zhao and Perry's project uses an ontology to represent and manage license agreements, ensuring both human and machine readability. The system automates tasks within the license management process through the SmArt framework, minimizing human intervention.

Risk Management in Enterprises: Durdu and Tutğaç employ a risk analysis methodology using one-on-one interviews and focus group meetings. Their approach is rooted in enterprise risk management, leading to the development of a SAM system.

Automated License Clearance: The DALICC framework relies on automated reasoning over licenses using ontologies and Answer Set Programming (ASP) to detect conflicts and ensure compatibility.

Blockchain Technology: Magnanini et al. utilize Ethereum-based smart contracts to manage licenses in a decentralized manner. This approach emphasizes secure, automated, and flexible license management, reducing the need for centralized oversight.

Proposed Solution: The License Manager system employs a more traditional relational database model to track and manage licenses, vendors, and products. It emphasizes data organization, ensuring each entity (vendors, products, licenses) is systematically recorded and managed, with relationships clearly defined through entity-relationship models (ERD).

6.3 Unique Contributions

Zhao and Perry (2008): Introduces the use of ontology for autonomic license management, offering a novel solution for making licenses machine-readable while maintaining human interpretability.

Durdu and Tutğaç (2022): Provides a risk-based approach to managing software licenses in large enterprises, highlighting the need for a SAM system and offering practical strategies for mitigating identified risks.

DALICC Framework: Presents a robust framework for automated license clearance, significantly reducing the complexity and cost of managing digital licenses through automated reasoning and conflict detection.

Magnanini et al. (2019): Offers a decentralized approach to software licensing, leveraging blockchain technology to create a trustless, automated system for license management, which is particularly innovative in reducing dependency on centralized systems.

Proposed Solution: The License Manager system distinguishes itself by focusing on a comprehensive yet straightforward approach to license management. It aims to integrate various aspects of license management (vendors, products, licenses, assignments) into a cohesive system that is easy to implement and use within an organizational context. It prioritizes practical usability and systematization over the more complex, automated, or decentralized approaches seen in the reviewed literature.

6.4 Comparison Of Effectiveness

Automation

Many of the reviewed projects emphasize automation, whether through ontologies, automated reasoning, or blockchain technology. These approaches aim to reduce manual intervention and increase efficiency, particularly in large-scale or complex environments.

Proposed Solution: While automation is a part of the License Manager system (e.g., automated tracking and reporting), the focus is more on systematic management rather than full automation. The system aims to be user-friendly and accessible, potentially sacrificing some of the advanced automation capabilities for simplicity and ease of use.

Scalability

The solutions discussed, such as DALICC and the blockchain-based system, are designed to be scalable across different environments and industries, particularly where large amounts of data or complex licensing models are involved.

Proposed Solution: The License Manager is likely more suited to small—to medium-sized enterprises that focus on organized, systematic management rather than handling extremely large or complex licensing environments.

Compliance and Risk Management

Compliance and risk management are central themes, particularly in Durdu and Tutğaç's work, which is explicitly focused on these issues. DALICC also addresses these concerns through automated conflict detection.

Proposed Solution: The License Manager includes compliance features but does not focus on risk management strategies as extensively. Instead, it provides tools for tracking and managing licenses, which can aid compliance but may require additional oversight.

Summary

The License Manager system proposed in the solution section of the document presents a more traditional and straightforward approach to license management, focusing on practicality and systematic organization. In contrast, the literature review projects explore more advanced, automated and in some cases, decentralized methods that aim to address the complexities of modern license management on a larger scale. Each approach has its strengths and is suited to different contexts depending on the specific needs, scale, and complexity of the license management tasks at hand.

CHAPTER SEVEN

CONCLUSION

In the rapidly evolving landscape of software development and distribution, effective license management has emerged as a critical challenge for organizations. The complexity of managing software licenses, particularly in environments where software is frequently updated and distributed across diverse platforms, necessitates innovative approaches that not only ensure compliance but also enhance operational efficiency. This thesis presents a novel approach to the development of License Management Systems (LMS), which addresses these challenges through a combination of advanced technologies and strategic methodologies.

7.1 Synthesis Of Findings

The proposed LMS framework integrates key concepts from autonomic computing, blockchain technology, and smart card-based security systems, drawing on the strengths of these technologies to create a robust and adaptable solution. As highlighted in the literature review, traditional license management systems often struggle with issues related to scalability, security, and user-friendliness. For instance, the study by Zhao and Perry (2008) emphasizes the importance of automating license management through ontology-based systems, which facilitate machine-readable interpretations of complex license agreements. Our approach builds on this by incorporating automation features that not only interpret licenses but also enforce compliance in real time, thus reducing the risk of human error and enhancing overall system efficiency.

Additionally, the integration of blockchain technology, as discussed by Magnanini et al. (2019), offers a decentralized approach to license management, ensuring that licenses are securely managed without the need for a central authority. This aspect of our system not only improves security but also allows for greater flexibility in

managing licenses across different jurisdictions and platforms, addressing a significant gap in current LMS solutions.

Moreover, the use of smart cards as a secure medium for license storage and transfer, as detailed by Aura and Gollmann (1999), provides an additional layer of security, ensuring that licenses cannot be easily tampered with or duplicated. This technology is particularly useful in scenarios where software distribution occurs both online and offline, allowing users to easily manage their licenses while maintaining the highest standards of security.

7.2 Contributions and Implications

The primary contribution of this thesis is the development of a comprehensive LMS that integrates these advanced technologies into a cohesive system. By leveraging autonomic computing for automated license enforcement, blockchain for decentralized license management, and smart cards for secure storage and transfer, our proposed system addresses the limitations of existing LMS solutions. It offers a scalable, secure, and user-friendly alternative.

The implications of this work are significant for both software vendors and users. For vendors, the system offers a more efficient way to manage licenses, reducing the administrative burden and minimizing the risk of non-compliance. The system provides users with a more transparent and secure way to manage their software licenses, ensuring that they can use the software they have purchased without worrying about complex licensing terms or potential legal issues.

Furthermore, this work opens up new avenues for research and development in the field of software license management. Future studies could explore the integration of additional security features, such as biometric authentication for smart card access, or the use of artificial intelligence to predict and mitigate potential compliance issues before they arise. Additionally, the scalability of the system could

be tested in larger, more complex environments, such as multinational corporations with diverse software needs and licensing agreements.

7.3 Summary

In summary, this thesis has presented a new approach to the development of License Management Systems, one that addresses the key challenges of scalability, security, and user-friendliness through the integration of cutting-edge technologies. By synthesizing findings from the literature with our own research and development efforts, we have created a system that not only meets the current needs of software vendors and users but also sets the stage for future innovations in this critical area of software development. As software continues to play an increasingly important role in both personal and professional contexts, the importance of effective license management will only grow, making this work both timely and essential.

This approach represents a significant step forward in the field of software license management, offering a solution that is not only effective but also adaptable to the changing needs of the software industry. As such, it provides a valuable foundation for future research and development, with the potential to shape the future of software license management for years to come.

CHAPTER EIGHT

FUTURE ENHANCEMENTS

The development of a robust software license management system, as proposed in this thesis, represents a significant advancement in addressing the complexities and challenges currently faced by developers and organizations. However, the dynamic nature of software development, coupled with the continuous evolution of licensing models and compliance requirements, suggests that there is ample opportunity for future enhancements to further refine and expand the capabilities of the system.

8.1 Integration of Advanced Automation and Artificial Intelligence

One potential enhancement lies in the integration of advanced automation and artificial intelligence (AI) techniques into the license management system. AI-driven tools could be employed to automatically analyze and interpret license agreements, flag potential compliance issues and suggesting optimal licensing strategies based on the specific needs of the project. For example, by utilizing natural language processing (NLP) algorithms, the system could parse complex legal language within license agreements and provide developers with simplified summaries or recommendations, reducing the risk of misinterpretation and non-compliance.

Moreover, AI could play a crucial role in predictive analysis, helping organizations anticipate changes in licensing needs as their software projects evolve. By analyzing historical data and trends, the system could forecast future licensing requirements, enabling proactive management and minimizing the likelihood of costly license violations. This would build on the existing automation features and further enhance the system's ability to manage licenses dynamically, in real-time, and with minimal human intervention.

8.2 Expansion to Blockchain-Based License Enforcement

While the current system incorporates blockchain technology for secure and decentralized license management, future enhancements could explore more advanced applications of this technology, particularly in the area of automated license enforcement. By utilizing smart contracts on a blockchain, the system could automatically enforce license terms without the need for manual oversight. For instance, a smart contract could be programmed to restrict access to software based on predefined conditions, such as the expiration of a license or the exceeding of usage limits. This would ensure that licensing agreements are adhered to with precision, reducing the risk of unintentional violations and enhancing overall compliance.

Additionally, blockchain's inherent transparency and immutability could be leveraged to create a comprehensive audit trail, providing organizations with a clear and verifiable record of all licensing transactions. This would not only facilitate compliance with legal and regulatory requirements but also offer valuable insights into software usage patterns, helping organizations optimize their software investments.

8.3. Enhanced User Experience With Smart Card Technology

The use of smart cards for secure license storage and transfer, as discussed in this thesis, provides a solid foundation for future enhancements aimed at improving user experience. One potential enhancement could involve the integration of biometric authentication methods, such as fingerprint or facial recognition, into the smart card system. This would add an additional layer of security, ensuring that only authorized users can access or transfer licenses while also streamlining the user authentication process.

Furthermore, advancements in smart card technology could enable the development of multi-function cards that store multiple licenses and credentials, reducing the need for users to carry and manage multiple cards. These multi-function cards could be integrated with mobile devices, allowing users to manage their licenses on the go through a dedicated mobile application. This would enhance the accessibility and convenience of the system, making it easier for users to comply with licensing requirements in a variety of settings.

8.4 Expansion of Licensing Models and International Compliance

As software continues to be developed and distributed on a global scale, the ability to manage a wide range of licensing models and comply with international regulations will become increasingly important. Future enhancements could include the expansion of the system's licensing database to cover a broader array of licensing models, including those specific to certain industries or regions. This would provide developers with the tools they need to navigate the complexities of international licensing, ensuring that their software complies with all relevant laws and regulations, regardless of where it is used or distributed.

Additionally, the system could be enhanced to support multilingual and multi-jurisdictional compliance, providing users with localized versions of license agreements and compliance tools. This would be particularly valuable for organizations operating in multiple countries, where licensing laws and regulations can vary significantly. By offering tailored compliance support for different regions, the system could help organizations avoid costly legal disputes and ensure that their software is used in accordance with local laws.

8.5 Integration With Emerging Technologies

Finally, the system could be enhanced by integrating with emerging technologies such as cloud computing, the Internet of Things (IoT), and edge computing. As these

technologies become more prevalent, they will introduce new challenges and opportunities in software licensing. For example, GenLM, as proposed by Dalheimer and Pfreundt (2009), offers a secure and robust framework for managing software licenses in grid and cloud computing environments, addressing the mobility and on-demand licensing requirements crucial for such distributed systems. IoT devices also often require specialized licenses due to their unique usage patterns and deployment environments. By integrating with these technologies, the license management system could provide tailored licensing solutions that address the specific needs of these emerging platforms.

Moreover, as cloud computing continues to evolve, the system could be enhanced to support hybrid and multi-cloud environments, where software licenses must be managed across multiple platforms and service providers. This would require the development of new licensing models and management tools that are specifically designed for the cloud, ensuring that organizations can maintain compliance while leveraging the full benefits of cloud technology.

8.6 Summary

In summary, while the current license management system represents a significant step forward in addressing the challenges of software licensing, numerous opportunities for future enhancements could further improve its functionality and effectiveness. By integrating advanced automation and AI, expanding blockchain-based enforcement, enhancing the user experience with smart card technology, broadening support for diverse licensing models and international compliance, and integrating with emerging technologies, the system could become an even more powerful tool for managing software licenses in an increasingly complex and dynamic digital landscape. These enhancements would not only help organizations stay ahead of licensing challenges but also enable them to fully realize the potential of their software investments.

REFERENCES

- Zhao, Q., & Perry, M. (2008, March). An ontology for autonomic license management. In *Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08)* (pp. 204–211). IEEE.
- Durdu, A., Tutğaç, A. (2022). Examination of Risks in Management of Software Licenses in Enterprises and Cost-Benefit Analysis. *Bilişim Teknolojileri Dergisi*, 15(4), 467 - 479. 10.17671/gazibtd.1019869
- Pellegrini, T., Havur, G., Steyskal, S., Panasiuk, O., Fensel, A., Mireles, V., ... & Schönhofer, A. (2019). DALICC: a license management framework for digital assets. *Proceedings of the Internationales Rechtsinformatik Symposium (IRIS)*, 10.
- Magnanini, F., Ferretti, L., & Colajanni, M. (2019, September). Efficient license management based on smart contracts between software vendors and service providers. In *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)* (pp. 1-6). IEEE.
- Vendome, C., Linares-Vásquez, M., Bavota, G., Di Penta, M., German, D. M., & Poshyvanyk, D. (2015, September). When and why developers adopt and change software licenses. In *2015 IEEE international conference on software maintenance and evolution (ICSME)* (pp. 31-40). IEEE.
- Di Penta, M., German, D. M., Guéhéneuc, Y. G., & Antoniol, G. (2010, May). An exploratory study of the evolution of software licensing. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1* (pp. 145-154).
- Noorian, L., & Perry, M. (2009, April). Autonomic Software License Management System: an implementation of licensing patterns. In *2009 Fifth International Conference on Autonomic and Autonomous Systems* (pp. 257-263). IEEE.

- Dalheimer, M., & Pfreundt, F. J. (2009, May). GenLM: license management for grid and cloud computing environments. In *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid* (pp. 132-139). IEEE.
- Gull, D., & Wehrmann, A. (2009). Optimized software licensing—combining license types in a license portfolio. *Business & Information Systems Engineering*, 1, 277-288.
- Aura, T., & Gollmann, D. (1999, May). Software License Management with Smart Cards. In *Smartcard*.