Democratic and Popular Republic of Algeria Ministry of Higher Education and Scientific Research Akli Mohand Oulhadj Of Bouira

Faculty of Science and Applied Sciences

Computer science department

Master thesis

In computer science

speciality: *Information Systems and Software Engineering*

Theme

Formal verification for an extended model

Supervised by

- DR. CHABANE DJEDDI
- DR. LYES BADIS

Submitted by

- MEZIANI Ahlam
- KHOBIZI Yasmina

2023/2024

Appreciation

Praise be to **ALLAH** through whose grace good things are accomplished. We thank GOD the Almighty, the Most Merciful, for giving us the strength, health, will, and courage to complete this modest work.

Firstly, we would like to thank our supervisors, **Mr. Badis Lyes and Mr. Chabane Djeddi**, for their patience, availability, and especially their wise advices, which has helped to fuel our reflection.

We would also like to express our gratitude to the friends and colleagues who provided us with their moral and intellectual support throughout our process.

We also thank the entire pedagogical team of the University of Bouira.

Finally, we extend our heartfelt thanks to the jury members who have agreed to evaluate our work.

Dedications

I dedicate this modest work :

To my dearest parents, Omar and Nora. No dedication can truly express my deep love, immense gratitude, and utmost respect for you. I will never forget the tenderness and love with which you have surrounded me since my childhood.

To my sisters Hiba, Aicha Manar, and Lina, for their unwavering support and encouragement. Your belief in me has been a constant source of strength and inspiration throughout this journey. Thank you for always being there for me.

To my friend Fares Saad, for his moral support and encouragement. Your steadfast friendship and unwavering belief in me have been invaluable. Thank you for always being there to lift my spirits and push me forward.

To my esteemed partner, Eda Mezian, for the countless nights we dedicated to laboring over this project together.

To my dear paternal grandparents: Mesaouda , Mohamed allah yrhamhoum et oum saad alah yarhamha .

To my dear maternal grandparents: Ali allah yrhamhou et Zohra allah yarhamha.

To all those I love and those who love me.

❤*k*hobizi Yasmina.

Dedications

Dedicated with love :

To my beloved and most kind person ever, my mother **Fatiha Aliouat**, for her prayers, advices, belief, and endless, unconditional love. You are my strength.

To my "always forever in mind ", father **Mohamed Meziani** (may he rest in peace) for believing in me once to make sure I become who I am today.

To my sisters, **Hayet** for her caring, **Fairouz** for her understanding, and **Leila** for her prayers. To **Samel, Nadia, Messaouda**,and **Saloua** for their love and support.

To my brothers, **Djaffer** my second father, **Omar** my constant supporter, **Karim** my peaceful place, **Mustapha**, and **Titah** my shining courageous star.

To my acolyte, **Rafif Slimani**, for being there through my ups and downs and supporting me no matter what.

To my Parabatai, **Thanina Lamraoui**, for always accepting me as I am and encouraging me with all her soul.

To my beloved and wise **Jasmine**, for her patience and understanding throughout our studies together. Thank you for being a sister and not just a binome.

To all my little confidants that I am so lucky to have: **Abdelkader(Nounou)**, **Doua**, **Sofiane**, **Khalil**, **Anes**,**Abd Elhay**, **Adem**, **Hiba**,**Maria** and **Idris** .

To my strong grandmother Melaid, whom I miss so much (may she rest in peace).

To my most fluffy creatures,**Alpha** and her mother **Violette**, for making my life more joyful.

To all the ones who kept believing in me and praying for me,I love you.

❤*Ahlam Meziani*

Abstract

GORE is an approach to requirements engineering that focuses on the intentions and objectives of various system stakeholders and the relationships between them, emphasizing the need for careful engineering and precision in defining system requirements. Many software projects fail due to inadequate requirements elicitation, particularly in the crucial and complex area of big data due to its specific characteristics.We identified the big data concepts that current(RE) methods do not support. Subsequently, we refine and adopt the KAOS method to ensure effective requirements elicitation.In this research, we introduce BKAOS, an extension of the KAOS method, designed to address big data characteristics such as (volume, variety..).First, we applied both KAOS and BKAOS to the same exemplary scenario.BKAOS exhibited promising outcomes, proving superior in eliciting big data project requirements.Next, to validate the integrity of BKAOS, we conducted formal proofs using Petri nets and Bigraphs formal verification analysis on both KAOS and BKAOS. Petri nets were applied to the four models and the meta-model of BKAOS, while Bigraphs were used to analyze the general case of both KAOS and BKAOS.

Keywords: Goal-Oriented Requirements Engineering, Requirements Engineering, Big data, KAOS, BKAOS ,formal verification.

Résumé

GORE est une approche de l'ingénierie des exigences qui se concentre sur les intentions et les objectifs des différentes parties prenantes du système et sur les relations entre elles, soulignant la nécessité d'une ingénierie minutieuse et d'une précision dans la définition des exigences du système. De nombreux projets de logiciels échouent en raison d'une élicitation inadéquate des exigences, en particulier dans le domaine crucial et complexe du big data en raison de ses caractéristiques spécifiques. Nous avons identifié les concepts du big data que les méthodes actuelles d'ingénierie des exigences ne prennent pas en charge. Dans cette recherche, nous introduisons BKAOS, une extension de la méthode KAOS, conçue pour traiter les caractéristiques des big data telles que (volume, variété…).Tout d'abord, nous avons appliqué KAOS et BKAOS au même scénario exemplaire.BKAOS a donné des résultats prometteurs, s'avérant supérieur dans l'élicitation des exigences des projets de big data. Ensuite, pour valider l'intégrité de BKAOS, nous avons effectué des preuves formelles à l'aide de réseaux de Petri et d'une analyse de vérification formelle Bigraphs sur KAOS et BKAOS. Les réseaux de Petri ont été appliqués aux quatre modèles et au méta-modèle de BKAOS, tandis que les Bigraphes ont été utilisés pour analyser le cas général de KAOS et de BKAOS.

Keywords: Goal-Oriented Requirements Engineering, Requirements Engineering, Big data, KAOS, BKAOS ,vérification formelle .

ملخص

تعد هندسة المتطلبات الموجهة بالأهداف (GORE) مقاربة لهندسة المتطلبات تركز على نوايا وأهداف الأطراف المعنية المختلفة بالنظام والعلاقات بينها، مشددة على ضرورة الهندسة الدقيقة والدقة في تعريف متطلبات النظام. تفشل العديد من مشاريع البرمجيات بسبب استخلاص غير كافٍ للمتطلبات، خاصة في المجال الحيوي والمعقد للبيانات الضخمة بسبب خصائصها المحددة. لقد حددنا مفاهيم البيانات الضخمة التي لا تدعمها طرق هندسة المتطلبات الحالية. في هذا البحث، نقدمBKAOS ، وهو امتداد لطريقةKAOS ، مصمم لمعالجة خصائص البيانات الضخمة مثل (الحجم، التنوع...). أولاً، طبقنا KAOS و BKAOSعلى نفس السيناريو المثالي. أظهرت BKAOS نتائج واعدة، وتفوقت في استخلاص متطلبات مشاريع البيانات الضخمة.

بعد ذلك، للتحقق من سلامةBKAOS ، قمنا بإجراء إثباتات رسمية باستخدام شبكات بتري وتحليل التحقق الرسمي Bigraphsعلى KAOS و BKAOSتم تطبيق شبكات بتري على النماذج الأربعة والنموذج الفوقي لـBKAOS ، بينما استخدمت Bigraphs لتحليل الحالة العامة لـ KAOS وBKAOS.

الكلمات المفتاحية :هندسة المتطلبات الموجهة بالأهداف، هندسة المتطلبات، البيانات الضخمة، KAOS، BKAOS، التحقق الرسمي.

Contents

List of Figures

List of Tables

Abbreviations list

RE	Requirements engineering
GORE	Goal Oriented Requirements Engineering
KAOS	Keep all objects satisfied
BKAOS	Big Data Keep all objects satisfied
ML	Modelling language
PN	Petri Nets

General introduction

Requirements Engineering (RE) is the initial step in the system development process, focusing on identifying stakeholders' objectives for the intended system. Its purpose is to define, analyze, and specify the needs and expectations of stakeholders for a computer system or software. The success of a project hinges on aligning system objectives with the real needs of users and the market, a task accomplished through RE. To achieve project success, it is crucial to thoroughly understand and appropriately document system requirements.RE addresses the challenges of system development by reducing the frequency and severity of problems through systematic methods and tools. These methods ensure a better understanding of system requirements, employing various approaches such as goal-oriented, scenario-based, or viewpoint-based strategies. The use of goals in RE is a common method to elicit high-level stakeholder concerns, including business goals, ensuring the quality of requirements by achieving these goals.The growing complexity of software applications demands methods capable of addressing intentionality. Goal-Oriented Requirements Engineering (GORE) is a sub-area of RE designed to meet these needs by enhancing traditional development approaches, whether they are object-oriented or aspect-oriented.In big data projects, where data is often massive, varied, and unstructured, RE is crucial for defining project objectives and expectations. It is essential for these projects to understand stakeholder needs, data sources, types of data to be analyzed, business objectives, and expected outcomes. BKAOS, an extension of the GORE approach KAOS, is specifically designed to meet the needs of big data projects. To improve the reliability of system analysis, techniques that combine the precision of traditional mathematical proofs with the efficiency of computer-assisted methods are necessary, minimizing human error. Graphical mathematical verification methods, based on

principles such as logic calculi, automate theory, and strong type systems, fulfill these criteria. To ensure the reliability of the BKAOS method, we conducted a study involving the formal verification of the extended model using two graphical mathematical verification methods: Petri nets and bigraphs.

Thesis Structure The remaining sections of these thesis are organized as follows:

- **Chapter 1:** we review the state of the art of RE and Big Data.
- **Chapter 2:** we delve into formal verification methods
- **Chapter 3:** presents a case study for KAOS and BKAOS
- **Chapter 4:** is the contribution chapter, where we conduct the formal verification of BKAOS.

Chapter 1

State of the art

1.1 Introduction :

For a development project to succeed, it is crucial to understand and properly document system requirements. Requirements Engineering (RE) employs various methods depending on the chosen approach. Goal-Oriented Requirements Engineering (GORE) addresses the intentionality of requirements by enhancing existing approaches, whether they are object-oriented or aspect-oriented, in the development of complex software applications.

This chapter will delve into the details of RE and the GORE approach, including its methods and the specific characteristics of big data. It will also emphasize the importance of applying RE practices to big data projects, highlighting their role in defining stakeholder needs, managing requirements, and ensuring the quality of the developed solutions.

1.2 Requirements Engineering :

1.2.1 Definition of Requirements Engineering :

In systems engineering, RE involves the process of developing and verifying system requirements. The primary objective of following good risk management practices is to ensure that the delivered system meets the customer's needs [3].

The RE process involves eliciting requirements, specifying requirements, analysis, verification and validation, and management [4].

This process is essential to ensure that the problem a client wants solved is clearly defined and transformed into a clear specification for a highly functional solution. This involves writing the requirement in technical language, using various models such as ER diagrams, data flow diagrams, function decomposition diagrams, and data dictionaries. After the requirement specifications are developed, they are validated to ensure that they are clear, correct, and well-defined.

1.2.2 RE Processes:

The RE process is a process that is iterative and consists of several steps, including:

1.2.2.1 Requirements Elicitation :

Learning, extracting, and discovering the needs of stakeholders is part of the initial sub-process of RE.

The requirements elicitation technique directs the requirements engineer towards the problem domain rather than potential solutions to those problems because its goal is to ascertain what problems need to be solved [4].

To elicit requirements, there are various techniques available. Among them, we have: [5]

- **Interviews:**Talking directly to stakeholders to understand their needs and expectations.
- **Surveys :** Stakeholders are given questionnaires to gather information about their needs and expectations.
- **Focus Groups :** A small group of stakeholders is convened to talk about their needs and expectations for the software system.
- **Observation :** To gather information about the needs and expectations of stakeholders in their work environment, this technique involves observing them.
- **Prototyping :** This method involves building a working model of the software to get feedback from stakeholders and confirm requirements.

1.2.2.2 Requirements specification:

Or the process of documenting the requirements, the goal of this step is to produce a formal software requirement models understandable by both the development team and the stakeholders.

Further information regarding the issue might be needed during specifying, which could again set off the elicitation process.This step covers all needs, both functional and non-functional, and the restrictions are all laid out by these models taken together.

1.2.2.3 Requirements analysing:

Development and congregation of the virtuous worthy requirements are the rudimentary needs of every organization to progress and produce quality software goods.

Requirements are thoroughly viewed, identified and then evaluated within the framework of the business requirements be helpful in future and will benefit the whole company and can also perceive that the recognized requirements should not be inconsistent [6].

To resolve any conflicts in the requirements, this step involves negotiating with different stakeholders.

To resolve these conflicting requirements, it's beneficial to give each of the stakeholders a set of priority points. As they see fit, they can allocate points between the conflicting requirements. The number of priority points received can determine the overall importance of any requirement [7].

1.2.2.4 Requirements verification:

It refers to the set of tasks that ensure the correct implementation of a specific function by the software [5].

1.2.2.5 Requirements validation:

The requirements need to be comprehensive, coherent, and precise to be verified in this step. In addition, it is necessary to ensure that the requirements are verifiable and meet the requirements and expectations of stakeholders [5].

During validation, it is useful to ensure that each requirement is attributed to a source. This allows requirements engineers to know who to contact for more information if needed [7].

1.2.2.6 Requirements management:

Requirements management involves associating requirements with different aspects of the software engineering process. This allows for easy identification and modification of requirements as the associated aspects change. This allows for easy identification and modification of requirements as the associated aspects change. It is important to ensure that all aspects of development linked to the requirements are modified accordingly.

The modified requirements can be examined to facilitate the propagation of changes throughout the project [7].

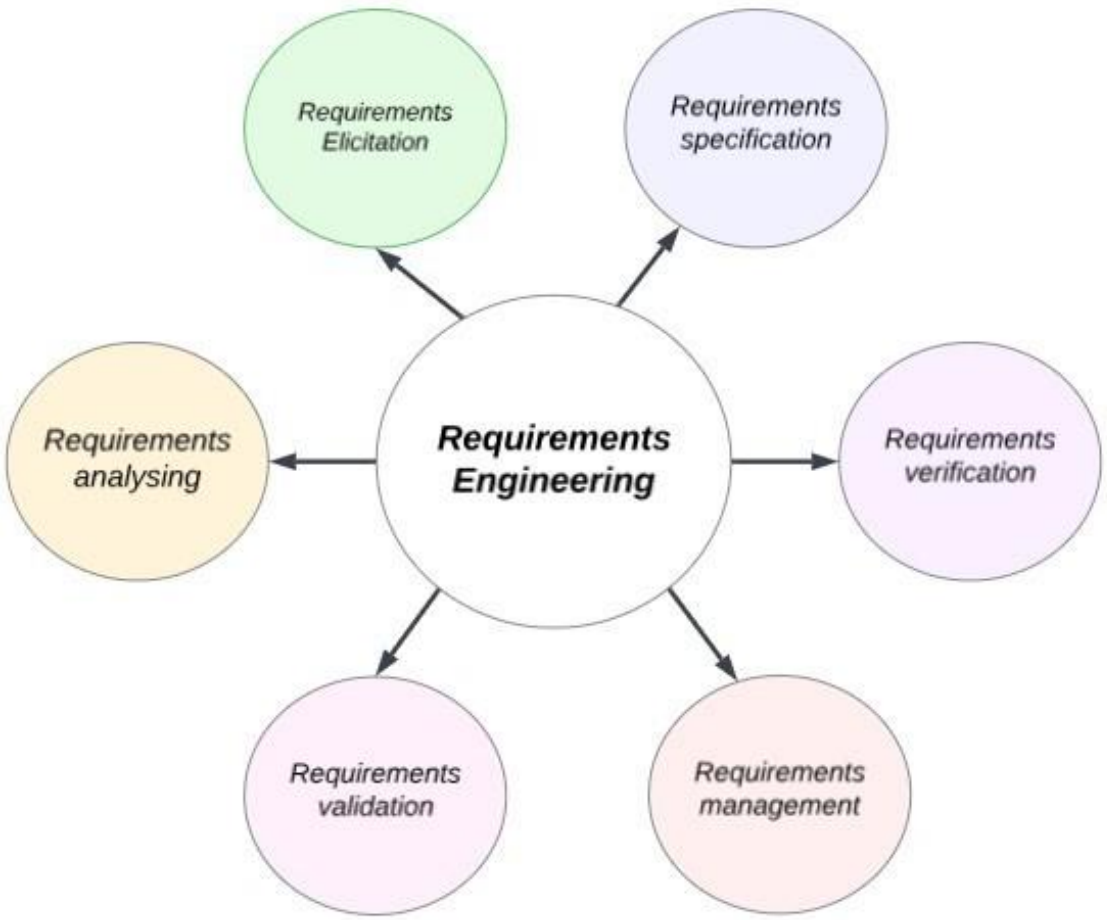

Figure 1.1: Requirements engineering steps

[8].

1.3 RE approaches:

There are a number of different approaches to requirements engineering in the literature. Each of them has a focus on specific activities in the requirements engineering process.

1.3.1 The scenario-based approach:

In this type of approach, requirements are described using scenarios. Scenarios are descriptions of the real world expressed using natural language, diagrams, etc [9] [10].

1.3.2 The GORE approach:

This is based on the definition of requirements as goals that can be divided and refined. Requirements This approach is described in more detail below [11].

1.3.3 The aspect-oriented approach:

This approach explicitly recognises the importance of properly identifying, processing and separating cross-cutting concerns that have otherwise been broken down into other requirements artefacts (use cases, goal models, etc.) [12] .

1.3.4 The ”Problem Frames” approach:

This is an approach for structuring the analysis of software requirements and designing a software solution. It helps to understand both the context in which the problem resides and the aspects relevant to the design of a solution. This approach focuses on functional requirements [13] .

1.4 Goal Oriented Requirements Engineering Gore:

Goal-Oriented Requirements Engineering involves studying and implementing goal models in Requirements Engineering. The language used to express a goal model is goal-oriented [14] .

Requirements are goals assigned to system agents, while expectations are goals assigned to environment agents.

Definition of Goals:

The goal is for the system to effectively detect errors through the cooperation of various agents within the software and its surrounding environment [15].

Definition of Agents:

Agents, including humans, devices, and software, are active components of the system and are responsible for achieving goals through cooperation.This may involve software components, external devices, and humans in the environment. The system being considered in the requirements engineering process is composite, as it includes both the software-to-be and its environment [16].

Most GORE methods typically involve the following main activities: goal elicitation, goal refinement, various types of goal analysis, and assigning responsibility for a goal to an agent. In this section, we present a variety of methods that adopt the GORE paradigm [17].

1.4.1 Istar:

IStar framework is an effective means for modeling and analyzing goals and interactions among social agents [18] .It is an ML used to model software at requirements level and has been extended to ft several specific application areas.

In the iStar framework, stakeholders are depicted as actors who rely on one another to accomplish goals, execute tasks, and provide resources. Each goal is evaluated from the perspective of its respective actor, resulting in a network of dependencies between pairs of actors. The iStar elements are categorized into Intentional Elements (including Goal, Softgoal, Task, and Resource), Actors (comprising General Actor, Role, Position, and Agent), and Links (such as Means-end, Decomposition, Contribution, and Actor Links). These elements are visualized in two distinct models: the Strategic Dependency (SD) model and the Strategic Rationale model (SR) .

The SD model outlines the connections and external dependencies between organizational actors.

]The SR model facilitates the analysis of how goals can be achieved through the contributions of various actors [19].

1.4.2 Agora :

Attributed Goal-Oriented Requirements Analysis Method

is characterized by the attachment of values of attributes (preference and contribution) to the goal graph thus providing goal quantitative analysis techniques.

During the process of refining and decomposing goals, an analyst assigns contribution values to the edges and preference values to the nodes of a goal graph. The contribution value of an edge indicates the extent to which a sub-goal contributes to its parent goal, while the preference matrix of a node (a goal) reflects the degree of preference or satisfaction of that goal for each stakeholder. These quantitative values assist the analyst in selecting among goal alternatives, identifying conflicts between goals, and assessing the impact of changes in requirements. Moreover, the values assigned to a goal graph and its structural characteristics enable the analyst to evaluate the quality of the resulting requirements specifications, such as their accuracy, clarity, and completeness.

The estimated quality values may suggest which goals should be improved and/or refined [20].

1.4.3 KAOS (keep all objects satisfied):

KAOS is a framework designed to elicit, specify, and analyze goals, requirements, scenarios, and tasks, as well as to assign responsibility [21].

A more detailed examination of this section will be provided in the subsequent chapter.

1.5 Big Data :

When we talk about manipulation of large volumes of data, about the vast amounts of data being generated at a high rate and in a variety of ways, we generally think of problems with data volume and speed of data processing.Here's a hint about big data .

1.5.1 Big Data Definition:

The evolution of the World Wide Web has transformed the types of data that need to be managed and tracked, the speed at which information flows into online systems, and the number of customers that companies must regularly handle. Due to these changes in the web environment, new definitions of big data have emerged, emphasizing technologies designed to manage this data. [22].

These definitions include :

1. According to the McKinsey Global Institute, "big data" refers to datasets so large that typical database software tools cannot effectively capture, store, manage, or analyze them. This definition is intentionally flexible and acknowledges that the threshold for what constitutes big data is constantly changing as technology advances. Therefore, big data is not defined by a specific size, such as a number of terabytes. Instead, it adapts over time with technological progress. Additionally, the definition can vary across different sectors, depending on the software tools available and the typical dataset sizes within a particular industry. Currently, in many sectors, big data ranges from a few dozen terabytes to multiple petabytes [23].

2. "Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making [24].

3. Big data are voluminous resources of information, with a high velocity and great variety that require innovative and cost-effective forms of information processing to improve understanding and decision-making [25].

4. Big Data, a term that refers to dataset that is so huge that performing efficacious analysis using the traditional methods becomes very difficult. Analysis of this data uses newly researched technologies and distributed architecture that makes extraction of value from the dataset possible. The term Big Data is a very big misnomer as it doesn't draw the attention to its various properties [26].

1.5.2 Big Data type of data:

The term ”big data” encompasses a multitude of sources, including sensors, intelligent objects, the Internet, and social collaboration technologies. These data sources are generally classified into three categories:

- **Structured data**

  The term ”structured data” refers to data stored in fixed fields within a file or record. This type of data is typically housed in a relational database management system (RDBMS) and can include both numbers and text. The sourcing of structured data can be done either automatically or manually, as long as it fits within the RDBMS structure [27].

- **Unstructured data**

  Unstructured data refers to any data that does not follow a predefined structure. While it may have an inherent internal structure, it lacks a consistent, predefined format. There is no specific data model, and the data is stored in its original format [27].

- **Semi-structured data**

  Semi-structured data falls between structured and unstructured data. It doesn't conform to the formal structure of a relational database but still utilizes tagging systems and identifiable markers to distinguish and make different elements searchable. While it doesn't fit the strict definition of structured data, it contains a self-describing structure, making it somewhat organized and easier to navigate [27].

1.5.3 Properties of big data:

The characteristics of big data are often highlighted by the 3Vs: volume, variety, and velocity. Volume refers to the immense scale of data, variety pertains to the diverse types of data, and velocity signifies the speed at which data needs to be processed. However, recent studies suggest that big data cannot be entirely encapsulated by just these 3Vs. Consequently, two additional dimensions, veracity and value, have been introduced to provide a more comprehensive understanding. Veracity emphasizes the reliability and accuracy of the data, while value underscores the significance and utility of the data in generating insights and driving decision-making processes.

Here's a good and simple overview for the five critical features of big data:

⇨ The sheer volume of data is a critical factor in big data analysis. It involves handling extensive amounts of unstructured, low-density data, which might include information whose value is uncertain, like machining conditions, material properties, or manufacturing control measures. While some organizations work with tens of terabytes of data, others deal with hundreds of petabytes [28].

⇨ Velocity refers to the rate at which the data are received, and possibly, to which some action is applied. The higher data-transmission rate is usually conducted directly to memory rather than written to a disc. Some intelligent products work in real time (or practically in real time) and require instantaneous evaluation and action [28].

⇨ The various types of data that are available. Conventional data types are structured and can be clearly organised in a relational database. With the rise of big data, data are presented in new types of unstructured data. Unstructured and semi-structured data types, such as text, audio or video, require additional pre-processing to obtain meaning and enable metadata. In addition, big data applications usually cope with sparse information [28].

⇨ Veracity refers to quality factor of the data which varies greatly. It caters to the question that whether the data which is being stored and mined is useful to the real life problem that is being dealt with.(Bigdata) Since one of three business leaders

don't trust the information they use to make decisions, establishing trust in big data presents a huge challenge as the variety and number of sources grows [29]

⇨ The value of the data refers to the usefulness of the data in relation to the purpose. The ultimate goal of the entire mega-data analysis system is to extract this value from the data. Data value is also related to the veracity or accuracy of the data. For some applications, the value also depends on the speed at which we can process the data [25].

Figure 1.2: Big Data properties [1].

1.5.4 Purpose of big data:

When organizations integrate big data into their business model, the primary concern often revolves around the tangible value it can bring. The data should be leveraged to enable more informed decision-making, optimize resource utilization, and improve the quality and efficiency of processes. Moreover, it should aid in precise customer segmentation, enhancing customer satisfaction, and fostering customer loyalty. Big data usage should also facilitate the creation of new business models, complementing existing product revenue streams while generating additional revenue from new products [22].

1.6 The impact of integration of RE in big data projects :

While the concept of big data has been present for many years, it's only in recent years that organizations have begun to grasp its potential for gaining insightful knowledge about their business operations. This understanding is empowering them to make more informed and effective business decisions [22].

A Big Data project involves the acquisition of Business analytics tools to perform advanced and predictive analyses of massive data.The predictive analysis of massive data is made possible by technological breakthroughs (in-memory, massive parallelisation) that make it possible to line arise the performance of predictive models based on cross-referencing internal and/or external data [30]

The integration of RE into big data projects is of paramount importance for the delivery of successful solutions that meet the needs of stakeholders, mitigate risks, optimise resource utilisation, and facilitate project adaptability and compliance. RE facilitates the gathering, analysis, and documentation of stakeholders' needs and expectations regarding big data solutions. This ensures that the resulting system aligns closely with what stakeholders actually require.

1.7 Conclusion:

In this chapter, we provided an overview of the field of requirements engineering, detailing its process and highlighting the GORE approach in particular.Furthermore, the practical importance of Big Data was explored, delving into its real-world implications and exploring the seamless integration of requirements engineering within it.

In the next chapter, we will discuss tow kind of Formal verification .

Chapter 2

Formal verification

2.1 Introduction :

Formal verification is a crucial process in system analysis, providing mathematical assurance that a system behaves as intended.

This chapter delves into two prominent formal verification methods to ensure the reliability and robustness of complex systems: Petri nets and bigraphs.

2.2 Formal verification:

The process of formal verification is the act of demonstrating the correctness of a system with respect to a specific formal specification or property [31].It involves the specification of the desired properties or behaviours of a system in mathematical terms, and then the systematic analysis of whether the system in question satisfies these properties.

2.2.1 Petri nets:

Carl Adam Petri named place/transition (PT) nets, a type of graphical mathematical modeling framework, for describing distributed systems. Petri nets are a promising tool for investigating and representing distributed, parallel, asynchronous, concurrent, nondeterministic, and stochastic information processing systems [32].

2.2.2 Graphical definition:

A Petri Network (PN) is a bipartite oriented graph value. It is comprised of two distinct types of nodes:

1. places, which are marked graphically by circles.
2. transitions, which are represented graphically by a rectangle or bar.

Each node in a network can be assigned a unique integer value, which is referred to as the ”marking.” These integers are referred to as ”tokens,” and are represented graphically by black dots.

The marking of a place node represents the number of tokens that occupy the node, whereas the marking of a transition node represents a quantity of tokens that are able to flow from the node into its input space. Alternatively, a transition with no input space is referred to as ”source transition,” whereas a transition with no output space is known as ”well transition.” places and transitions are interconnected via oriented arcs, where:

- Each arc connects either a place to a transition or a transition to a place,but not between two places or two transitions.
- Each arc is labeled with a numeric value, referred to as a weight.

It is possible to think of an arc with a weight of k as a collection of k parallel arcs.

An arc lacking a weight is equivalent to an arc with a weight of 1.

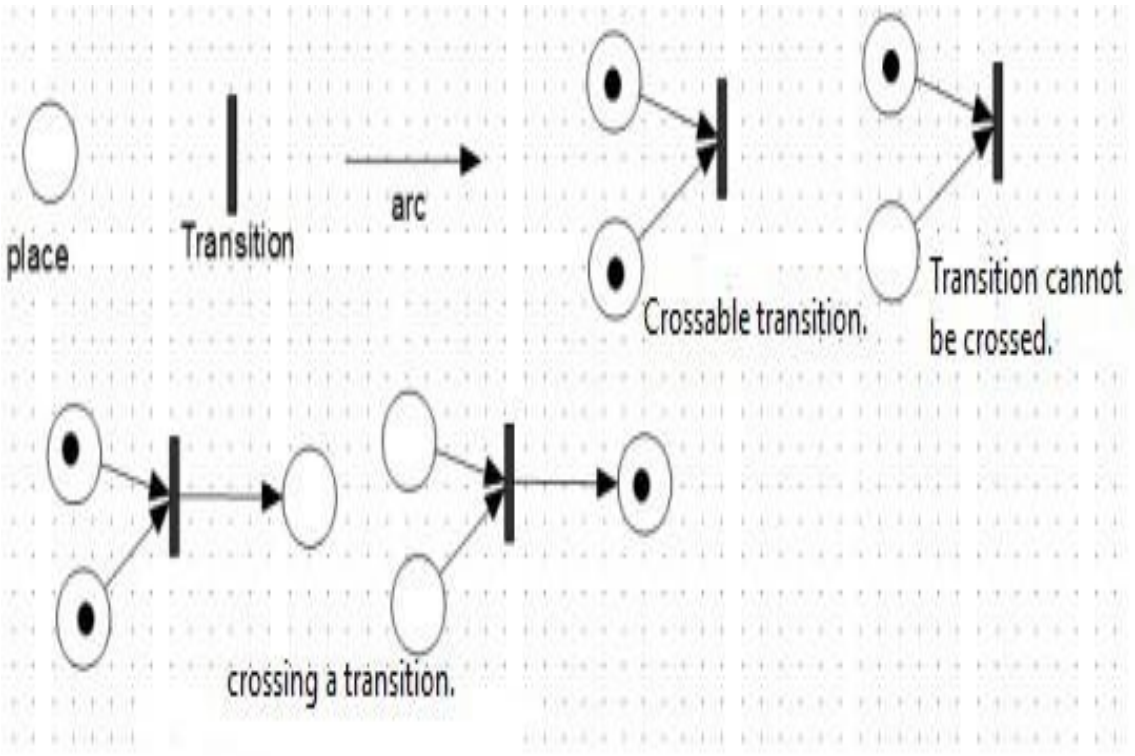

Figure 2.1: Symbols and basic principles of PN

[32].

2.2.2.1 PN Definition :

The foundation of the Petri-Net theory is a rule that must be understood: the transition enabling and firing rule. A specific type of directed graph known as a Petri net is comprised of an initial state known as the initial marking M_0 [32].

A Petri net's underlying network N is a directed, weighted, bipartite graph made up of two different types of nodes known as:

- The places P_i that make it possible to describe the states of the modeled system. All of these places is denoted $P = \{P_1, P_2 \ldots\}$
- The transitions (Ti) which represent the changes of states. All of these transitions is denoted $T = \{t_1, t_2 \ldots\}$ [33]

Places and transitions are connected by oriented arcs where arcs are either from a place to a transition or from a transition to a place (never directly between places or between transitions).Each arc is assigned a weight (integer number) ,by default this number is equal to 1.

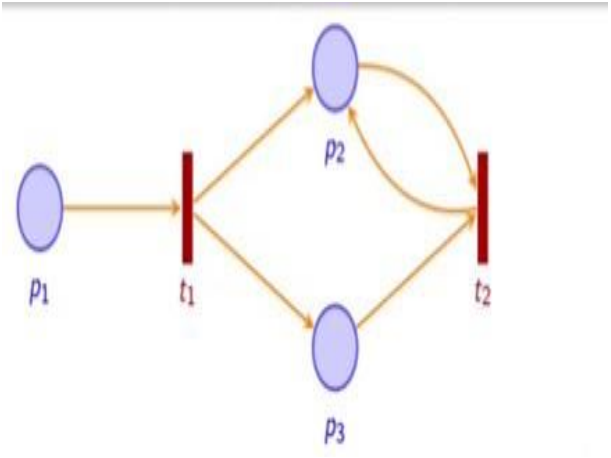

Figure 2.2: Petri net graph

2.2.2.2 Formal Definition:

A Petri net graph is a 5-tuple (P, T, W-,W+,M_0) where

- P, T are finite, non-empty, disjoint sets.
- P is a finite set of places, $P = \{P_1, P_2, P_3\}$..
- T is a finite set of transitions, $T = \{t_1, t_2, \}$...
- W- : P × T → N the forward incidence function.
- W+ : P × T → N the backward incidence function.
- $M_0 \subset N_p$ is the initial marking of the network [34].

W-(p,t) : is the precondition associated with transition t and place p. It defines the number of marks (tokens) which must be present in p for t to be traversable. If t is crossed, this number of token is removed from p.

W+(p,t) : is the post-condition associated with transition t and place p. It defines the number of marks (tokens) which are generated in p when t is traversed .

If, F is the set of arcs then : $F \subset (P \times T) \cup (T \times P)$ with :
$F = \{(P_1, t), (t, P_2), (t_1, P_3), (P_2, t_2), (P_3, t_2), (t_3, P_2)\}$ and
$(P \cap T) = \emptyset$ and $(T \cap P) = \emptyset$

A transition t is said to be enabled if each input place p of t is marked with at least w(p,t) tokens, where w(p,t) is the weight of the arc from p to t . An enabled transition may or may not fire (depending on whether or not the event actually takes place) [35].

A firing of an enabled transition t removes w(p,t) tokens from each input place p of t , and adds w(p ,t) tokens to each output place p of t , where w (t,p) is the weight of the arc from t to p [35].

2.2.2.3 The marking of places:

In each place, there is either a positive or zero integer number of marks or tokens. If a place Pi has a number of marks, it will be referred to as either M(Pi) or mi.

At time i, Mi the marking in the network is determined by the vector of these markings $Mi = (m_1, m_2, \ldots, m_n)$.

M_0 is the initial marking that represents the initial state of the modeled system [36].

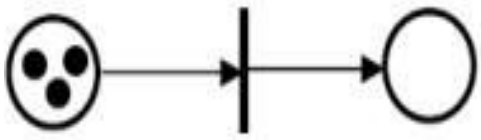

Figure 2.3: Initial marking on a PN.

2.2.2.4 The Crossing of a Transition :

Petri nets include a formalism that allows one to transition from one marking to another in order to account for the evolution of the system being modelled. The formalism for moving from one marking to another is the crossing of a transition [33].

The franchissement (or the termination) of a transition can only be accomplished if each of the places is in a suitable state. The transition contains enough tokens to equal the weight of the arc [33].

2.2.2.5 PN application example :

In Figure 1.6 a transition rule is illustrated using a will known chemical reaction : $2H_2 + O_2 \Rightarrow 2H_2O$

In Figure l.6(a), two tokens in each input position indicate that the transition t is enabled and that two units of H_2 and O_2 are accessible. The marking will change to the one seen in Figure l.6(b) when t has fired, at which point the transition t is disabled [34].

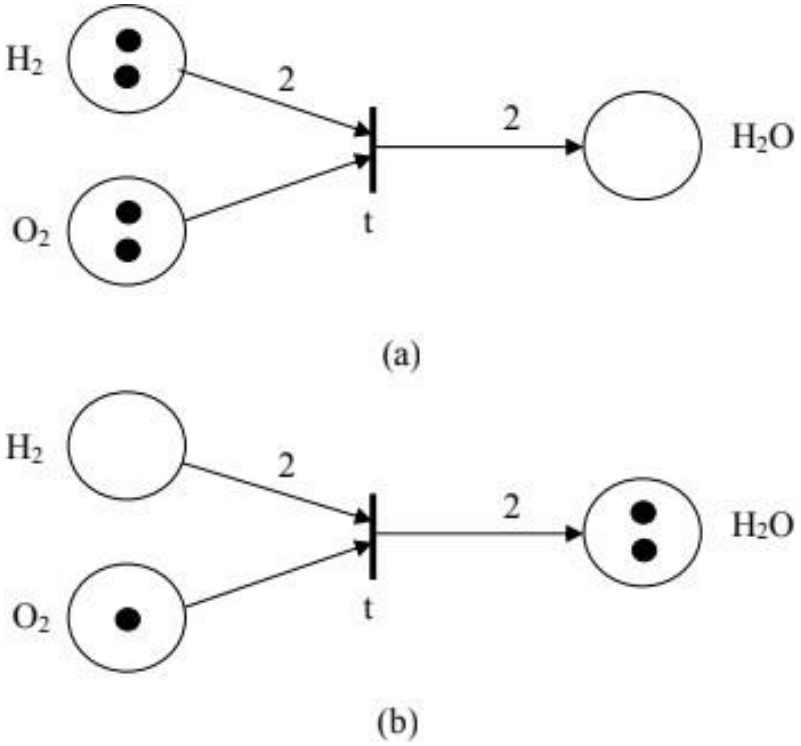

Figure 2.4: An illustration of transition firing rule [34].

- (a) Marking the transition t before it is enabled and fired.
- (b) The indication that t is disabled following a firing.

2.2.2.6 Characterization of PN:

A lively network:

From the initial marking, which represents the starting point of a process, it is important to note that all transitions in the network are always traversable, regardless of the system's evolution. If this is not the case…

This means that a portion of the proposed model becomes unmarkable after a certain point in the system's evolution, rendering it inactive. In such cases, the model is considered partially useless beyond that point. To address this, the system must be reinitialized

to include this sub-PN. Any network designer will want to ensure the system remains functional [37].

Bornitude of the network:

If the total number of markings in all of the petri net's places is finite, the net is said to be bounded.

This property ensures that the number of parts in a stock does not increase without bound. The concept of boundedness is important when analyzing the P-invariant or place invariant [37].

Mutual exclusion:

Physical systems frequently involve mutual exclusion, such as an on/off position or the presence or absence of an object at a particular location.

This principle of mutual exclusion is represented in the Petri net model by the exclusive presence of a token in one place, to the exclusion of one or more other places. The model utilizes a token to manage the availability and sharing of resources such as machines, robots, and stock.

The token can rotate within the model to facilitate resource sharing [37].

Resetting:

Resetting means ensuring that the system can return to its initial state. This is often essential as it guarantees the possibility of returning to the initial conditions of a piece of equipment, regardless of its state of evolution. The model representing this system should include a clear and concise explanation of the reset process. This problem is addressed by studying T-invariants or transition invariants [37].

2.2.3 Different types of PN:

- **qualitative PN:** distinct space – molecular level (token count).

- **stochastic PN:** discrete space: transitions start to happen after a random variable-determined probabilistic wait.
- **continuous PN:**Ordinary differential equation in continuous space for every place (concentration).
- **hybrid PN:** integrates the characteristics of stochastic and continuous PN (for instance, reactions with high rates are regarded as continuous and reactions with low rates as stochastic).
- **coloured PN:** It makes it possible to describe recurring interactions in a spatial setting.

2.2.4 Bigraphs:

2.2.4.1 Definition:

Milner introduced Bigraphs as a universal modelling language [38]. As part of a new graphical formalism for creating, modeling, and evaluating ubiquitous computing systems, Bigraphs and corresponding reactive systems have been created. From a structural perspective, a Bigraph is a graphical meta-model that highlights the mobile systems' proximity and connection [39].

The theory's two primary goals were to unify existing theories by creating a general theory that included several current calculations for mobility and competition. The first goal was to be able to incorporate the aspects into the same formalism as key ubiquitous systems [40].

2.2.4.2 Bigraphs Constituents:

A bigraph is formed by combining two independent structures: the square graph and the link graph. A set of nodes shared by both graphs represents the physical or virtual entities of a mobile code-distributed application. Place graphs capture mobile location, while link graphs capture mobile connectivity.

Basic signature Controls which are not active (including the atomic controls) are called passive. A signature K is a set containing elements known as controls. Each control in

K is associated with a natural number called its arity, denoted as (k,ar). The signature K also specifies which controls are atomic and identifies the active controls among the non-atomic ones. Controls that are either passive or atomic are referred to as passive controls [41].

In the bigraph, each node has a control that means the name or identity of an element in the specified system. For instance, the basic signature of the **bigraph B** shown in Figure 2.5 is:

$$K = \{v0 : 1, v1 : 1, v2 : 0, v3 : 2, v4 : 2, v5 : 1\}$$

Interface A bigraph has interfaces, which define its use as a construction block.

Place graph The place graph is used to specify the notion of locality (nesting of nodes) in a big graph. It consists of a forest composed of several trees , whose root is always a region to which several leaves (nodes or sites) are hierarchically attached [42].

Link graph A link graph is used to describe the connectivity (interaction between nodes) of a big graph [42].

The outgoing and incoming interfaces (n, m) of the places graph, respectively represented by its roots and sites, are noted by $\{0, 1...n-1\}$, they are disjoint from its nodes. Roots (n) can be parents of nodes and sites (m), but there is no parent for them; the sites may be the threads of the roots and knots but there is no son for them.

The outgoing and incoming interfaces (X, Y) of the link graph are normally sets of names: respectively its outgoing (X) and incoming (Y) names.

For example, we choose the outgoing interface 2 = $\{0, 1\}$ providing the distinct roots as the parents of the nodes v0 and v4; for the incoming interface, we choose 1, that is, it has one site 0. We write the graph of places like: $B^p : 1 \longrightarrow 2$; We choose for this bigraph example, the outgoing and incoming interfaces of the link graph: and x. The graph of links is: $B^L : \{x\} \longrightarrow \emptyset$ Thus, the bigraph $B : (n, X) \longrightarrow (m, Y)$, which does not have open links in our example, is noted: $B : (1, x) \longrightarrow (2, \emptyset)$.

The bigraph interfaces purpose is to allow the construction of (more complex) bigraphs from (simpler) bigraphs, and to consider a bigraph as a substructure of another. Indeed, we can conceive bigraphs from smaller bigraphs through their interfaces and algebraic operations on the bigraphs, as composition and tensor product operations [43].

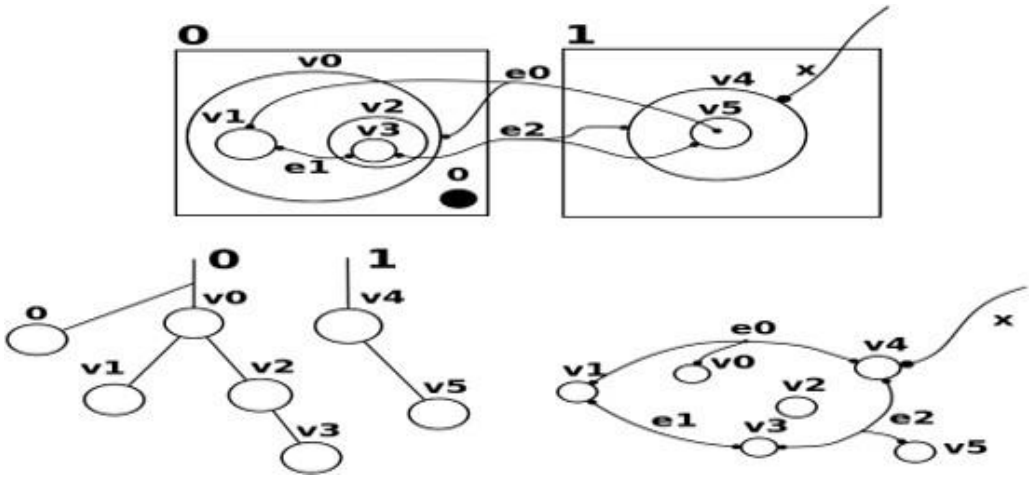

Figure 2.5: Simple Bigraph B and its associating places and links graphs

2.2.4.3 Formal definitions:

Bigraphs: a bigraph is defined by $G = (V, E, ctrl, G^P, G^L) : I \rightarrow J$, such that :

- V is a finite set of nodes that can be nested.
- E is a finite set of hyper-arcs (edges).
- $ctrl : V \rightarrow \kappa$ is a transformation which associates with each node v_i of V a controller $k \in \kappa$ indicating the number of ports and its dynamic behaviour.
- $G^p = (V, ctrl, prnt) : m \rightarrow n$ is the graph of places associated with G, where $prnt : m \cup V \rightarrow V \cup n$ is a parent function associating each node with its hierarchical parent. m and n respectively represent the number of sites and regions in the place graph.
- $G^L = (V, E, ctrl, link) : X \rightarrow Y$ is the link graph of G, where $link : X \cup P \rightarrow E \cup Y$ is a transformation showing the flow of data from internal names X or ports P to external names Y or arcs E.
- $I =< m, X > etJ =< n, Y >$,are respectively the internal and external interfaces of the graph G, with :
  - m is the number of sites, i.e., locations in the graph likely to host to host new elements,
  - X is the set of internal names,
  - n is the number of regions, i.e. elements of the bigraph that can be integrated into other bigraphs,
  - Y is the set of external names.

We call G the combination of its constituents G^p and G^L, writing $G =< G^p, G^l >$.

An interface $I =< 0, Ø >$, can be denoted by $I = \xi$ [44].

2.3 Conclusion:

In summary, formal verification is essential for ensuring the accuracy and reliability of complex systems. By employing PN and bigraphs, we can thoroughly analyze system

behavior, identify potential issues, and model intricate interactions. This comprehensive approach provides a robust verification framework.

In the contribution chapter, we will apply these two formal verification methods to the BKAOS method.

Chapter 3

Case Study of KAOS and BKAOS

3.1 Introduction:

The process of requirements elicitation involves the gathering, identification, and documentation of requirements from a variety of sources, including stakeholders, users, and other pertinent parties. The KAOS GORE approach is employed as the methodology for achieving this process.

This chapter will provide a comprehensive examination of the KAOS GORE approach and its associated sub-goals, with a particular focus on their application to big data projects. Additionally, it will introduce a novel method known as BKAOS, which has been specifically developed for the purpose of capturing all essential requirements in big data projects.

3.2 KAOS(keep all objects satisfied):

Known as Knowledge Acquisition Automated System , it's a method called Goal-Oriented Requirements Engineering (GORE), widely employed for requirements elicitation. This method is recognized for its multi-paradigm approach, enabling the integration of various levels of expression and rationale. It utilizes semiformal techniques for modeling and organizing goals, qualitative methods for evaluating alternative selections, and formal approaches for handling critical elements [45] .

The KAOS requirements model is made up of four sub-models strongly linked by consistency rules :

- **Goal Model:**

  Defines the goal of the system and its surroundings. The hierarchical organization of this model involves refining higher-level goals (strategic goals) to lower-level goals (requirements) [17] .

- **The Object Model:**

  Despite the potential benefits of using KAOS, it remains challenging to construct a glossary within this framework. However, KAOS does permit Analysts for glossary work in a progressive and concurrent manner during the definition of goals and requirements.

  This is achieved by developing a KAOS object model, which encompasses objects, agents, entities, and the relationships between them [2] .

  This one describes the domain's vocabulary. A UML class diagram represents it [17] .

- **Responsibility Model:**

  The KAOS approach permits the construction of responsibility diagrams from the underlying model, which display the full scope of requirements and expectations associated with a given agent.

  Aside from goals, agents represent a significant element of the KAOS conceptual framework and can be responsible for meeting requirements and expectations, either through humans or automated components.

The deployment of the KAOS methodology necessitates the association of individual requirements or expectations with a specific agent, or agent group, responsible for their fulfillment. This process is fundamental to the success of KAOS, as it ensures the alignment of agent-specific responsibilities with the broader objectives of the system under analysis.

A requirement is a low-level goal that a software agent is expected to achieve. It falls under the responsibility of the software agent.

The system environment includes an expectation, which is a goal that an agent is expected to achieve [2] .

- **Operational Model:**

The KAOS operational model outlines the necessary behaviors for an agent to meet its requirements. These behaviors are manifested through operations carried out by the agent. These operations manipulate the objects defined in the object model, such as creating objects, initiating object state changes, and triggering additional operations via events sent and received [2] .

It represents system operations in terms of their individual characteristics and how they relate to goals, objectives, and responsibilities [17] .

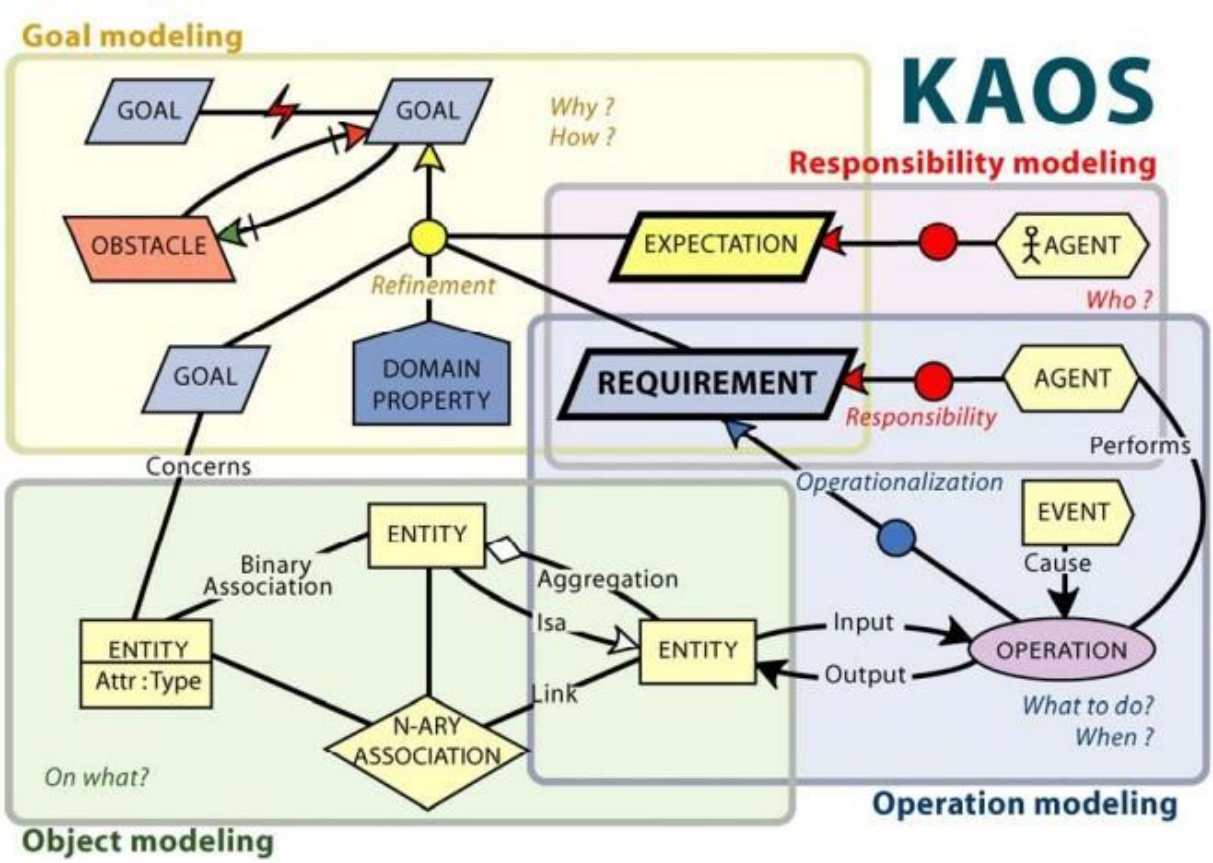

Figure 3.1: the four perspectives of KAOS Models [2]

3.2.1 KAOS glossary:

- **Agent:**

  In order to accomplish objectives, an active entity, referred to as a processor, executes tasks. Agents can be analyzed as a complete entity or in segments, and they can also originate from the software's surrounding environment. Additionally, human agents are included within this environment.

- **Association:**

  The definition of an object is contingent upon the existence of other objects that are linked by an association.

- **Conflict:**

A goal is considered to be in conflict with another goal if, under certain boundary conditions, it is not possible to achieve both goals simultaneously.

- **Domain Property:**

  Objects within the software's environment can be described in two ways: as domain invariants or as hypotheses. A domain invariant is a property that is universally true for every state of a domain object, such as a regulation or physical law. On the other hand, a hypothesis is a property that is believed to be true for a specific domain object.

- **Entity:**

  Autonomous objects are defined as entities that exist independently of other objects.

- **Environment:**

  A segment of the universe that can engage with the software being examined.

- **Event:**

  An instantaneous object, that is to say, an object that exists in a single state, initiates operations performed by agents.

- **Expectation:**

  Within the environment, an agent is assigned a goal.

- **Requirement:**

  The goal is to assign a goal to an agent of the software under investigation.

- **Goal:**

A prescriptive assertion is a statement that captures an objective to be met through the cooperation of agents. It prescribes a set of desired behaviors. Requirements and expectations are goals.

- **Obstacle:**

A condition that is not a goal and may prevent achieving certain goals, which defines a set of undesirable behaviors.

- **Operation:**

Specifies the state transitions of objects, both those that are the input and output of an operation, as well as those that act as agents within the operation itself.

Operationalisation:

A relationship exists between a requirement and operations. When the intent is constrained, each execution of the operations will entail the requirement, which is the basis for this relationship. The expected properties (goals) and behaviors (operations) are connected by this relationship.

Refinement:

A relationship exists between a goal, which we may refer to as the ”main goal,” and the sub-goals that contribute to the fulfillment of the main goal. Each of the sub-goals is a sufficient condition for the satisfaction of its respective main goal. The conjunction of all the sub-goals must be a sufficient condition for the main goal.

Responsibility:

When an agent is given the responsibility of meeting the linked requirement, it is defined as the relationship between an agent and a requirement [2].

3.2.2 The example illustrative scenario for KAOS :

In this example, we consider the case of a company wishing to improve customer satisfaction and increase product sales by implementing a project based on massive data and data-driven strategies to improve the shopping experience.Several issues have been identified, including the development of personalized product recommendations for customers.This will be achieved by analyzing their purchase history, browsing behavior, and demographic informations.Analyzing customer data in real-time to offer instant recommendations during the shopping experience.Optimize inventory management by analyzing sales data, demand forecasts, and supply chain metrics to ensure availability of popular products while minimizing excess stock and stockouts.By analyzing interactions with customer service, social media channels, and online reviews, we can enhance customer service.Design and execute targeted marketing campaigns by analysing customer segmentation data, market trends and advertising effectiveness indicators. This will ensure that the campaigns reach the right audience.To streamline the payment process, it is essential to analyse user experience data, website traffic patterns and conversion rates.

3.2.3 The application of KAOS Model on the example :

3.2.3.1 The Goal model :

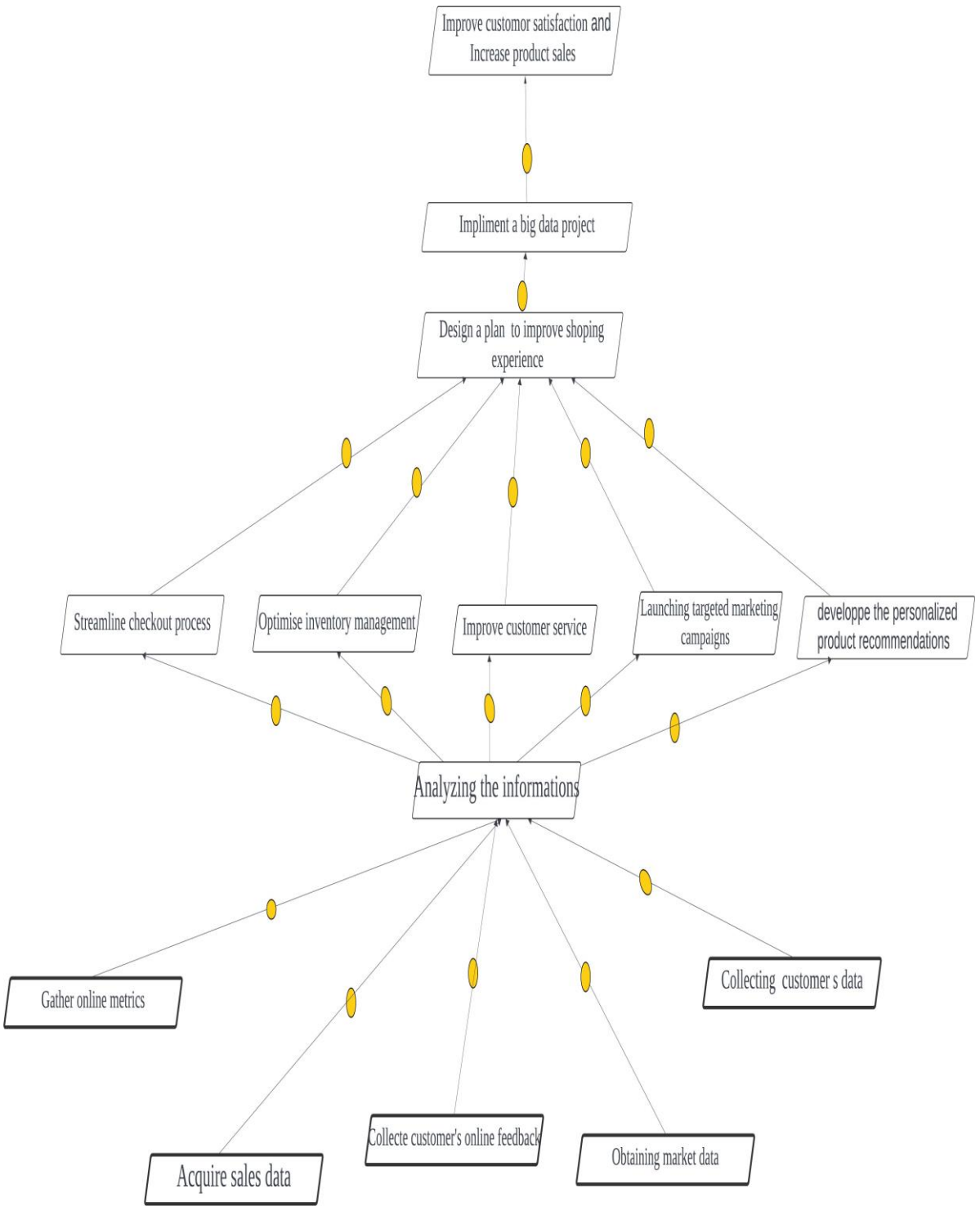

Figure 3.2: the application of KAOS Goal model

The goal is represented by each parallelogram in the figure. Refinements of a parent goal and a list of sub-goals are communicated through yellow circles. The diagram can be understood in the following manner: Our principle goal to **Improve customer satisfaction and increase product sales** at the same time for that we have to make another goal which is implementing a big data project base on other goal (Design a plan to improve shopping experience) to reach the last goal we have five sub-goals to attempt first (streamline checkout process, optimize inventory management, improve customer service, launching targeted marketing campaigns and develop the personalized product recommendations) to reach these sub-goals an analyze of the information must be done and for that we need five requirements

3.2.3.2 The responsibility model:

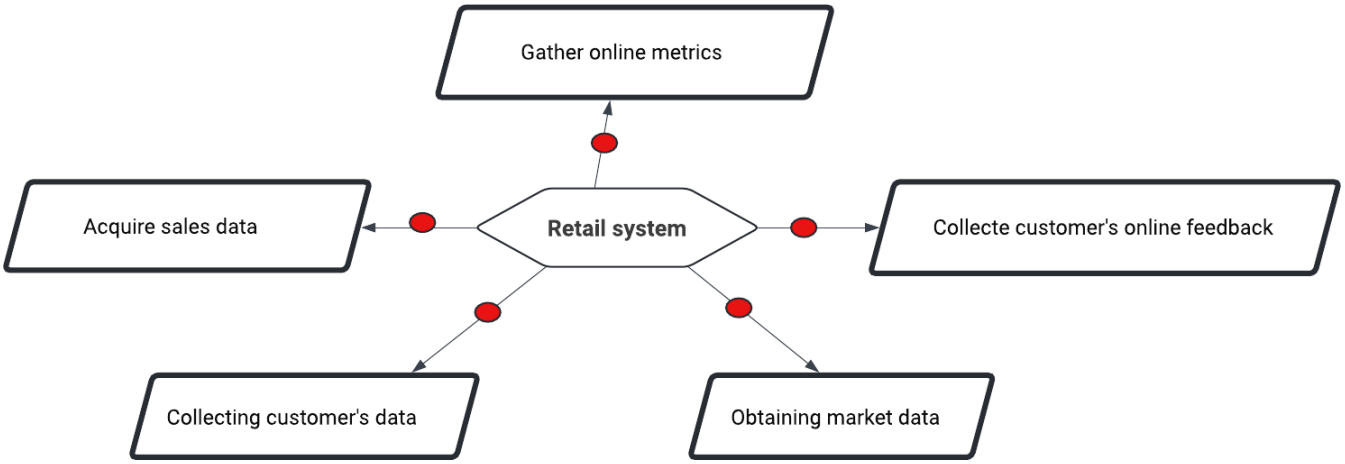

Figure 3.3: the application of KAOS responsibility model

A responsibility diagram outlines the expectations and requirements that each agent is expected to meet or has been assigned.To build a responsibility diagram a review is made on the different requirements and expectations in the goal model, we assign an agent to each requirement.For instance the **figure 3.3** has been updated as :

- the (Retail system) is responsible of the five requirements (Gather online metrics, acquire sales data, collecting customer's data, obtaining market data and collect customer's online feedback) .

3.2.3.3 The Operational model:

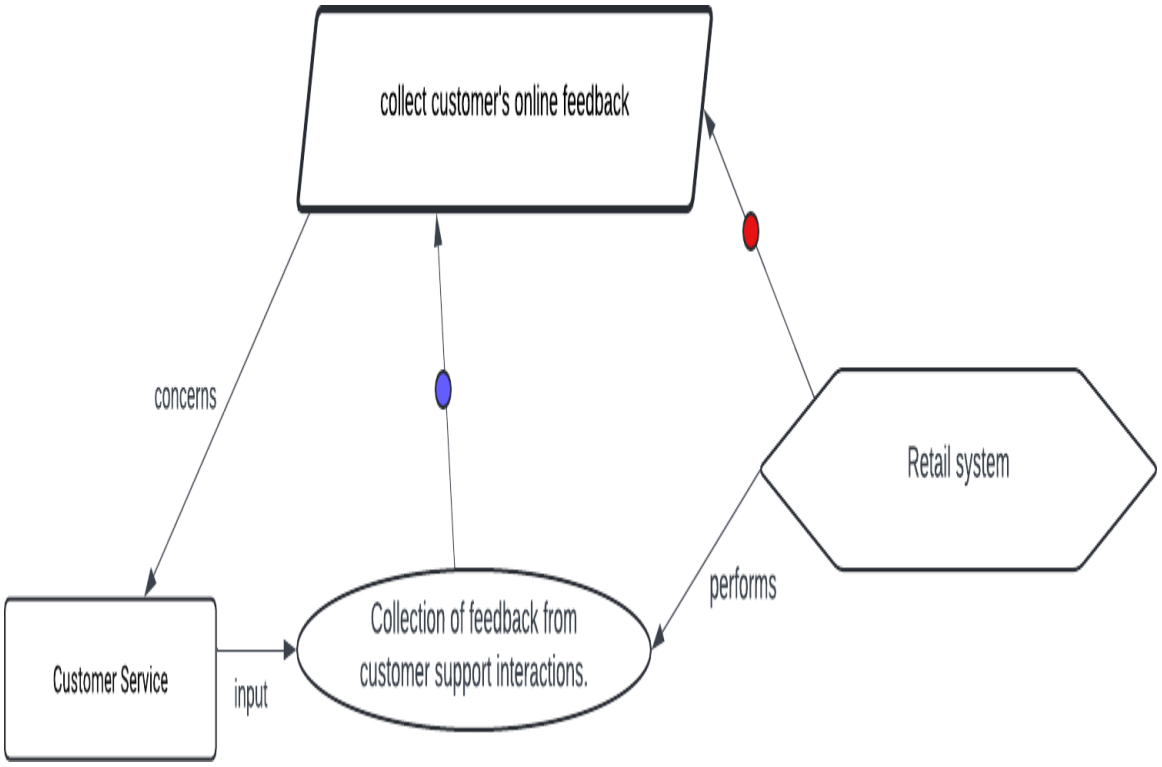

Figure 3.4: the application of KAOS Operational model

the operation model describes all the behaviors that agents need to fulfill their requirements.

Figure 3.4 shows how an agent (retail system) is responsible for a requirement (collect customer feedback). The agent performs an operation (collecting feedback from customer support interactions) and there is a link between the operation and the requirement.

3.2.4 The Object model:

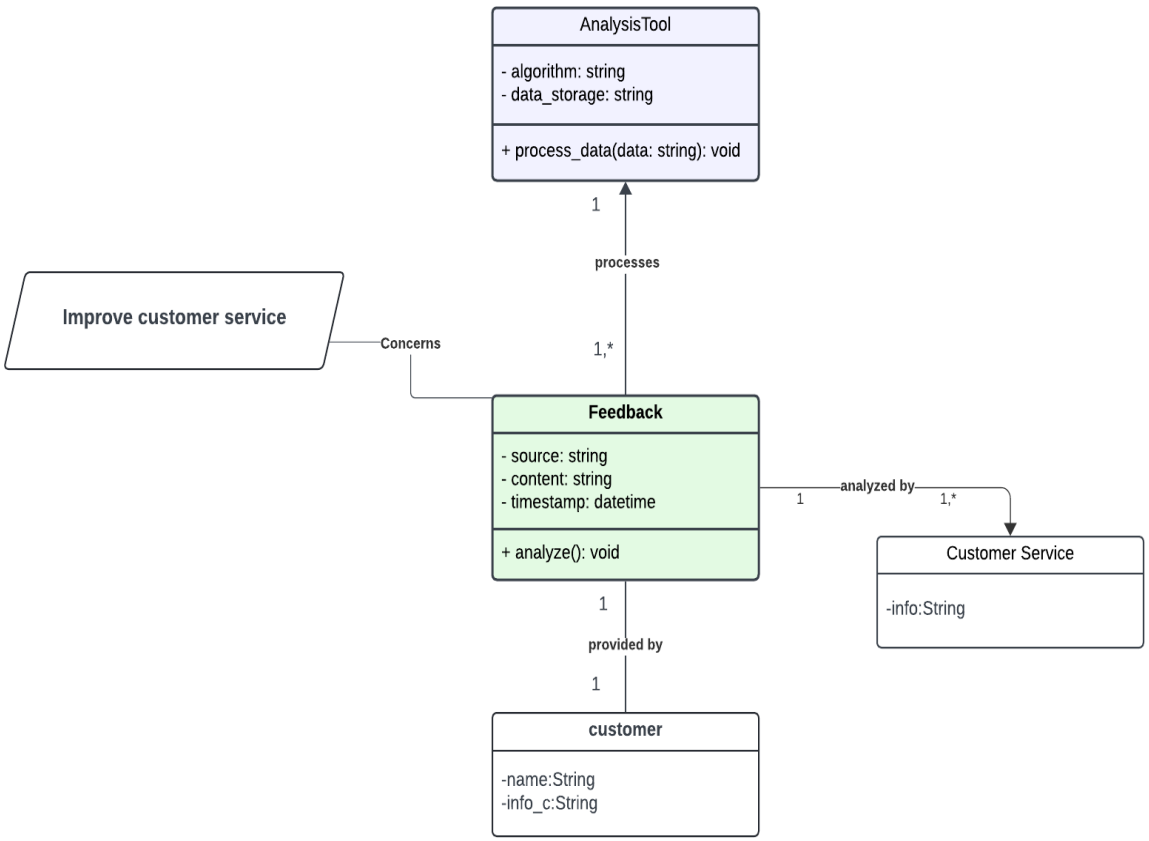

Figure 3.5: the application of KAOS Object model

The figure **Figure 3.5** represents a KAOS object model diagram, which is used to describe the objects, their attributes, operations, and the relationships between them in the context of improving customer service through feedback analysis.

3.2.5 Kaos meta-model:

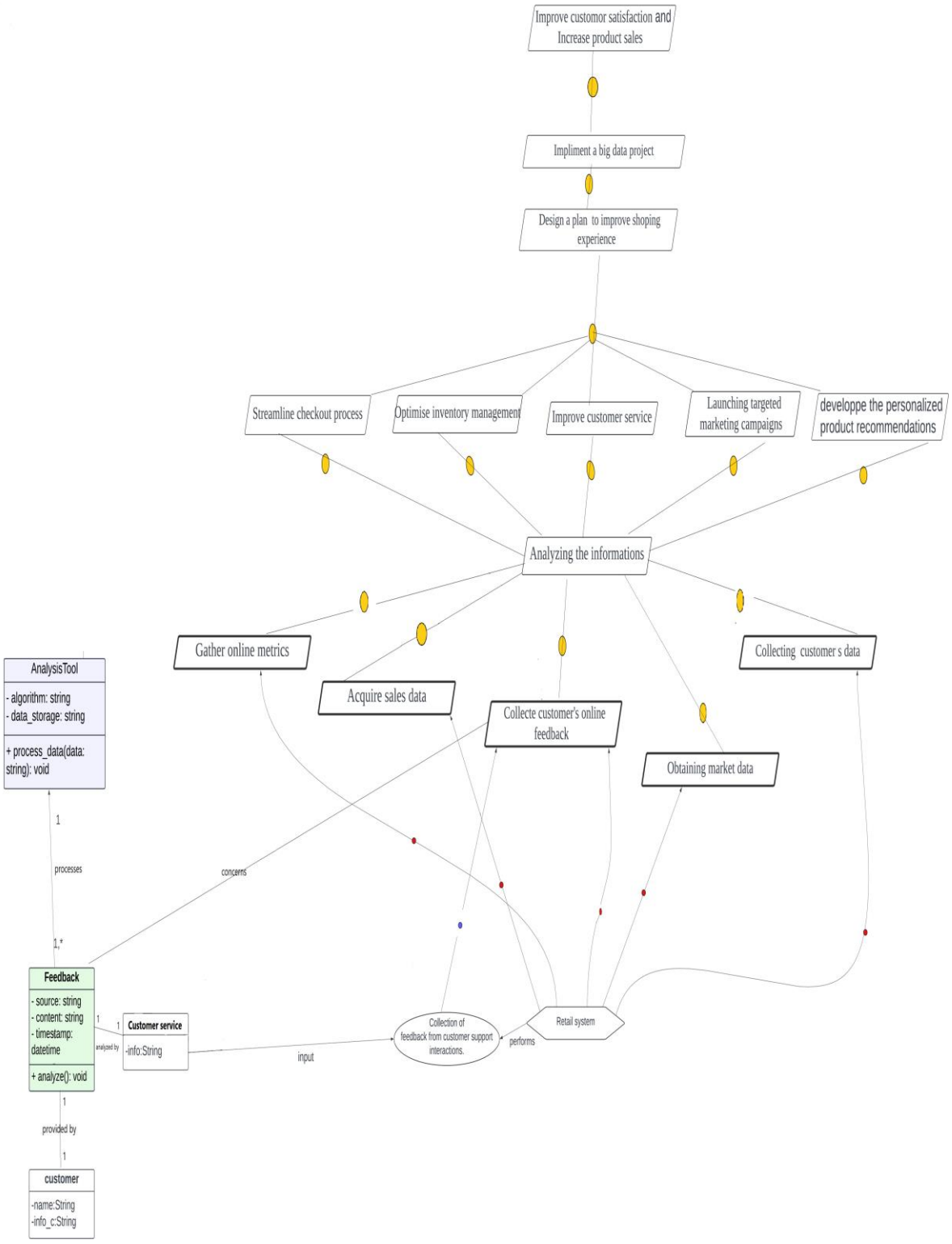

Figure 3.6: Kaos meta model

Figure 3.6 show the KAOS meta-model which is the combination of the other four models .

3.2.6 BKAOS:

An extension of the KAOS framework, known as BKAOS, was created to encapsulate all the criteria needed for Big Data projects.The developer introduced several principles to Kaos to enable large data projects, which present unique difficulties, by increasing the project's accuracy.These encompass the ideas of goal durability, processing volume, variation, and execution time [46] .

3.2.6.1 The Concepts added to KAOS :

The volume of data to process:

the volume is a crucial point in Big data projects they corresponds to the mass of information produced every second and the amount of data to be managed.This issue cannot be supported by the current RE methods An independent concept that gathers the requirements for big data projects. It seems evident to specify the volume of data to process.

The execution time :

Big data is becoming increasingly important because of the significance of its results. RE approaches must take into account the time it takes to execute them. The execution time must be accurate, otherwise the late result is considered incorrect.

The variety of data:

The utilization of big data technology allows for the examination and comparison of different types of data, such as conversations, social media messages, and photos from various sources. These distinct elements highlight the versatility that Big Data offers. When conducting requirements engineering, it is crucial to account for the characteristics of the data being analyzed.

The durability of a goal:

Engineers typically create systems that can function during and after a specific time. That may no longer be useful, as it is currently the case in big data projects. So, the RE method for Big data must specify the durability of a goal from the Beginning. The characteristics mentioned above are not allowed by KAOS Complete and refined elicitation of the requirements for Big data. BKAOS made it to Overcome these issues and enable a superior elicitation.

3.2.7 The example illustrative scenario for BKAOS :

We maintain the same meaning as described in section 3.2.2 .In BKAOS new concepts are added to each goal :

The goal **"Collecting customer's data"** must be donne within one week by analyzing 1TB of structured data nature, and it must be in operation during the shopping experience .

The goal **"Gather Online metrics"** must be donne within one week by annalyzing 20GB of structured data nature, and it must be in operation during the shopping experience .

The goal **"Acquire sales data"** must be donne within two weeks by analyzing 100GB of structured data nature, and it must be in operation during the shopping experience .

The goal **" Collect customer's online feedback "** must be donne within one month by analyzing 10GB of unstructured data nature, and it must be in operation during the shopping experience .

The goal **"Obtaining market data "** must be donne within three weeks by analyzing 50GB of unstructured data nature, and it must be in operation during the shopping experience .

BKAOS gives more completeness and refinement to the requirements eliminating any ambiguous or unclear specifications

3.2.8 The application of BKAOS Models on the example :

As it is mentioned in **section 3.2.7** the concepts of big data were added to each model.We keep the same description as the last description in The application of KAOS Models for each diagram in this application ..

3.2.8.1 The Goal model :

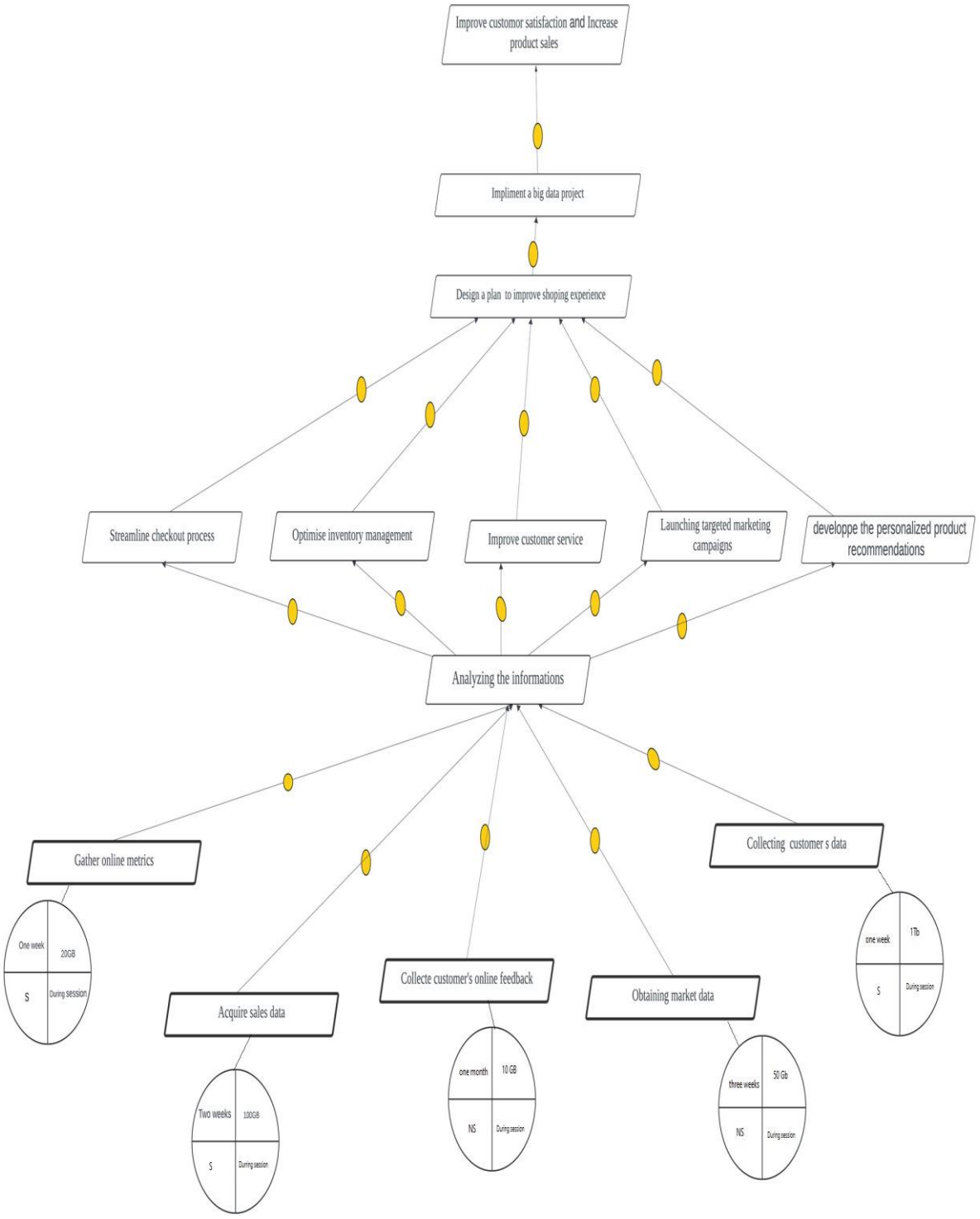

Figure 3.7: the application of BKAOS Goal model

The goal model diagram represents a hierarchical structure of objectives aimed at improving customer satisfaction and increasing product sales. At the top, the primary goal is to ”Improve customer satisfaction and Increase product sales.” To achieve this, the next

step is to ”Implement a big data project,” which requires ”Designing a plan to improve the shopping experience.” This central goal is broken down into five sub-goals: streamlining the checkout process, optimizing inventory management, improving customer service, launching targeted marketing campaigns, and developing personalized product recommendations.

These sub-goals are supported by the key requirement of ”Analyzing the information.” To effectively analyze information, the system needs to fulfill five critical requirements: gathering online metrics, acquiring sales data, collecting customer feedback, obtaining market data, and collecting customer data. Each of these requirements has specific big data characteristics associated with it, such as the volume of data (e.g., 20GB for online metrics, 100GB for sales data, etc.), the frequency or duration of data collection (e.g., one week, two weeks, etc.), and the session specifics, ensuring the system can handle large, diverse, and timely datasets efficiently.

3.2.8.2 The responsibility model:

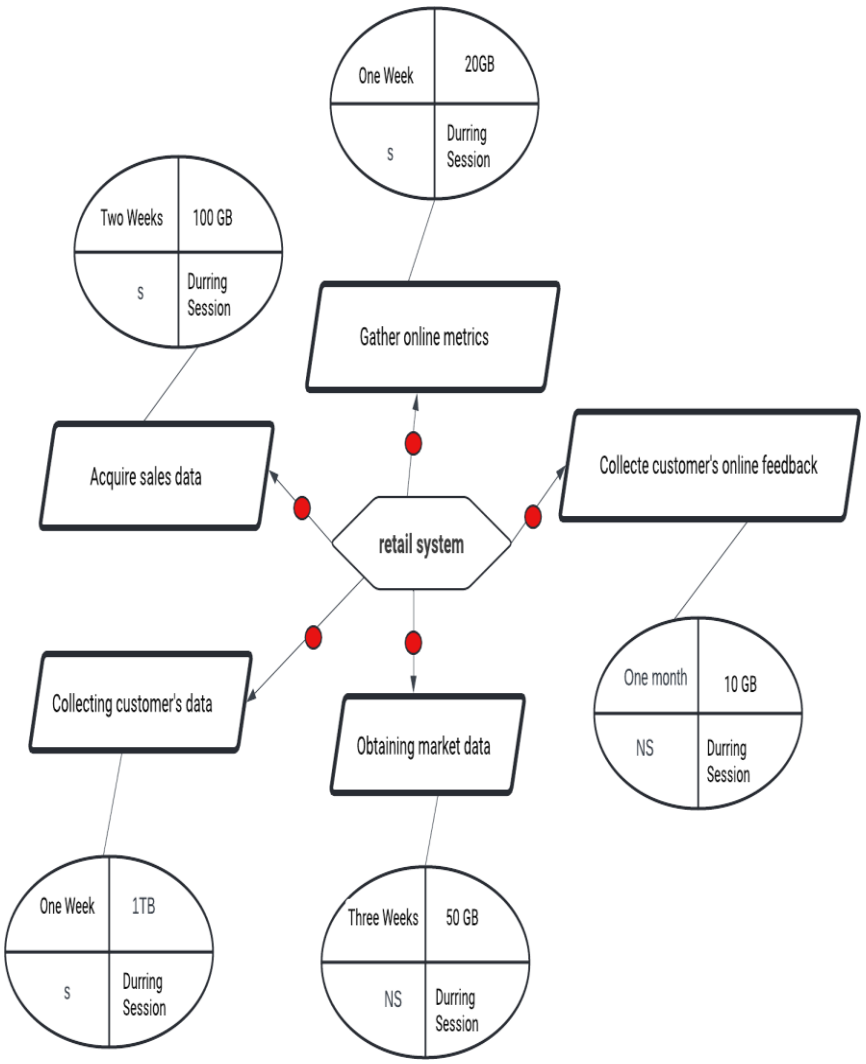

Figure 3.8: the application of BKAOS responsibility model

Figure 3.8 The responsibility diagram assigns the retail system to handle five key requirements: gathering online metrics, acquiring sales data, collecting customer data, obtaining market data, and collecting customer online feedback. Each of these requirements is linked to the retail system, indicating its responsibility. Integrating big data requirements, the system must handle large volumes of data, ensure real-time processing, and maintain data accuracy and reliability. This involves efficiently managing vast amounts of customer information, sales figures, and market trends while providing timely and accurate feedback to improve overall service and decision-making.

3.2.8.3 The operational model

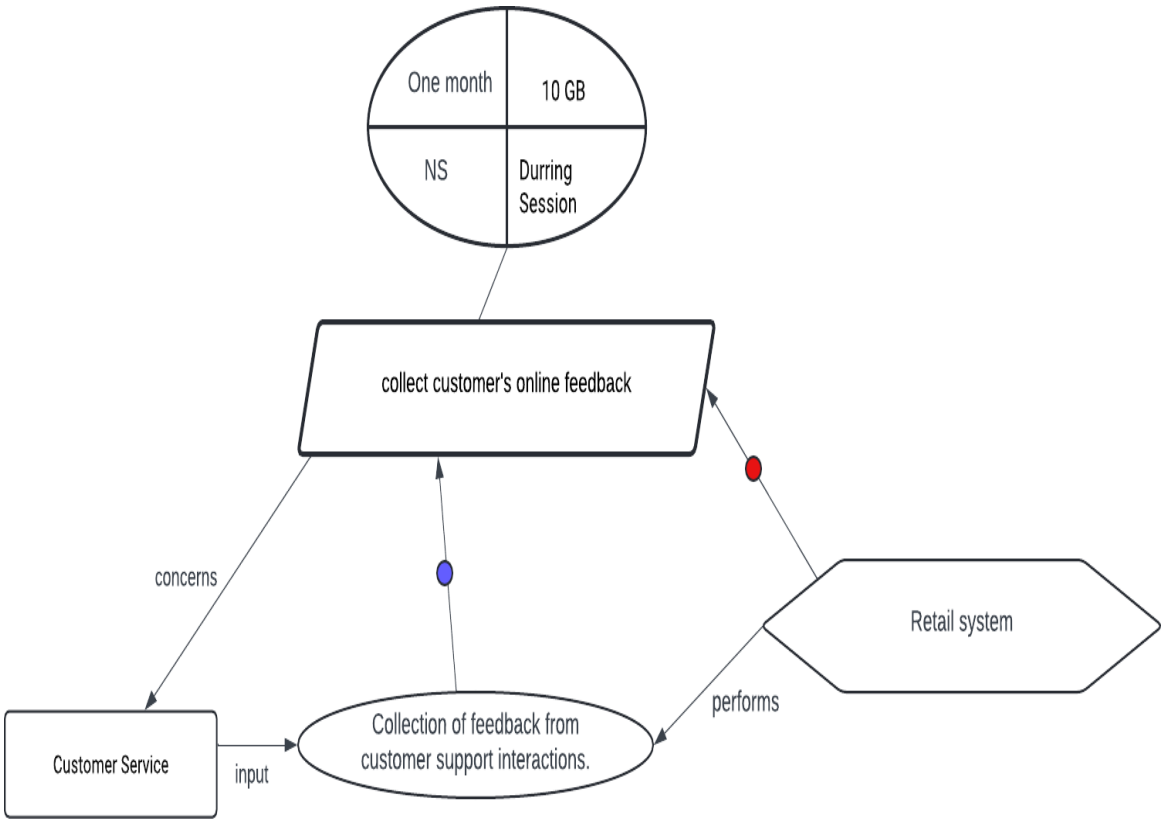

Figure 3.9: the application of BKAOS operational model

The BKAOS operation model diagram (**Figure 3.9**) shows the retail system's responsibility for collecting customer feedback by performing operations that gather input from customer support interactions. The process involves customer service agents inputting data, which the system collects and verifies at various points (indicated by blue and red dots). Big data characteristics are integrated, with the volume represented by "10 GB," velocity by "During Session," and veracity ensured through verification points. The goal is to gather valuable online feedback efficiently to enhance customer service.

3.2.8.4 BKAOS meta model:

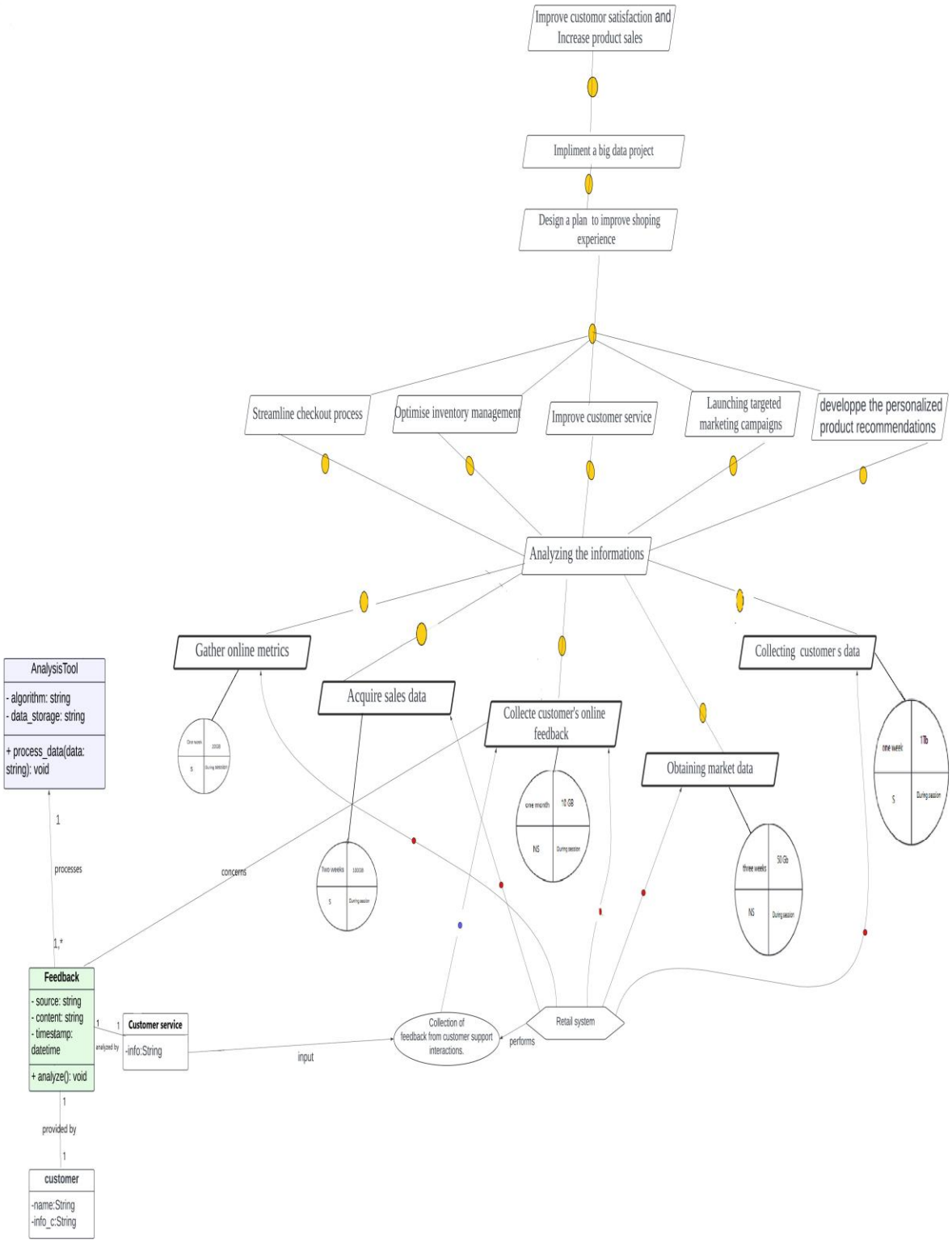

Figure 3.10: Bkaos meta model

Figure 3.10 shows the KAOS meta-model which is the combination of the other four models .

3.3 Conclusion:

In the preceding chapter, we conducted a comprehensive analysis of the KAOS GORE approach and its sub-goals. Furthermore, our understanding was enhanced by integrating the concepts of Big Data, which were pivotal in the development of BKAOS. This novel methodology was devised with the specific objective of enhancing the precision and efficacy of big data projects.The next chapter will discuss the formal verification models .

Chapter 4

Contribution

4.1 Introduction:

Formal verification techniques often involve formal logic, such as temporal logic or predicate logic, to express system properties and model checking or theorem proving to verify these properties. The goal of formal verification is to provide strong guarantees about the correctness of a system.

In the next chapter we will apply tow formal verification techniques to demonstrate the correctness of BKAOS .

4.2 Petri Nets Formal Verification:

4.2.1 Application of the PN in the general case of KAOS:

The general case of KAOS through a PN will be studied in this section.

Set of places p:

$P = \{P_{goal}, P_{subgoals}, P_{Agent}, P_{\text{Requirements}}, P_{Entity}, P_{Operation}, P_{Operationalization}, P_{Expactation}, P_{\text{Domain Property}}\}$

P_{goal}:representing the place of the goal .

$P_{subgoals}$:representing the place of the sub-goals .

P_{Agent}:representing the place of the Agent .

$P_{Requirements}$:representing the place of requirements.

P_{Entity}: representing the place of the entity.

$P_{Operation}$: representing the place of the Operation.

$P_{Expectation}$: representing the place of the Expectation.

$P_{DomainProprety}$: representing the place of the Domain Property .

$P_{Operationalization}$:Representing the place of the Operationalization .

Set of transitions:

$T = T_{Refinement}, T_{Responsibility}, T_{Performs}, T_{Concerns}, T_{Relationshipsbetweenentities}$

$T_{Refinement}$:Representing the transition of the Refinement .

$T_{Responsibility}$:Representing the transition of the Responsibility .

$T_{Performs}$:Representing the transition of the Performance .

Data input token

Data collection token

Introduction of the incoming arc function I(t,p):

This function denotes the number of incoming arcs from transition t to place p.

Introduction of the outgoing arc function O(t,p):

This function indicates the number of outgoing arcs from transition t to place p .

Initial Marking Function M_0

The number of tokens in each place p represented by M_p.

4.2.2 Application of the PN in the general case of BKAOS:

In this section, we will study the general case of BKAOS by a PN.

Set of places p:

$P = \{P_{goal}, P_{subgoals}, P_{Agent}, P_{\text{big data requirements}}, P_{Entity}, P_{Operation}, P_{Operationalization}, P_{Expactation}, P_{\text{Domain Property}}\}$

P_{goal}:representing the place of the goal .

$P_{subgoals}$:representing the place of the sub-goals .

P_{Agent}:representing the place of the Agent .

$P_{Requirements}$:representing the place of the requirements.

$P_{big_data,equirments}$:representing the place of the big data requirements.

P_{Entity}: representing the place of the Entity.

$P_{Operation}$: representing the place of the operation.

$P_{Expectation}$: representing the place of the Expectation.

$P_{DomainProprety}$: representing the place of the Domain Property .

$P_{Operationalization}$:Representing the place of the Operationalization .

Set of transitions:

$T = T_{Refinement}, T_{Responsibility}, T_{Performs}, T_{Concerns}, T_{Relationshipsbetweenentities}$

$T_{Refinement}$:Representing the transition of the Refinement .

$T_{Responsibility}$:Representing the transition of the Responsibility .

$T_{Performs}$:Representing the transition of the Performance .

Data input token

data collection token

Introduction of the incoming arc function I(t,p):

This function denotes the number of incoming arcs from transition t to place p.

Introduction of the outgoing arc function O(t,p):

This function indicates the number of outgoing arcs from transition t to place p .

Initial Marking Function M_0

The number of tokens in each place p represented by M_p.

4.2.3 The simulation of each model and the metamodel of BKAOS :

In our simulation, we employ the WoPeD (Workflow Petri Net Designer) tool.

WopeD is a tool for drawing, managing, simulating, and analyzing workflow process definitions using ”workflow nets.”

semantic analysis:

- Qualitative analysis:

Structural analysis: The model has been subjected to a comprehensive examination, which has encompassed a series of rigorous checks pertaining to its structural integrity. These include network statistics, the correct utilisation of operators, the absence of any violations pertaining to free choice, the presence of all requisite components, and the model's overall structural soundness.

- Soundness:

Workflow net property: The model represents a valid workflow network.

Initial marking: The initial configuration of the tokens is correct.

Boundedness: The network is bounded, which means there is no infinite growth of tokens.

Liveness: The network is alive, which means that it is always possible to achieve the final goal from any state reached.

4.2.3.1 the simulation of the goal model:

Structural Analysis:

Net Statistics:

Wrongly Used Operators: 0

Free-choice Violations: 0

S-Components: Wellstructuredness is confirmed.

Soundness:

Workflow Net Property: The net adheres to the workflow net property.

Initial Marking: The initial marking is correct.

Boundedness: The net is bounded.

Liveness: The net is live.

The qualitative, structural, and strength analyses of were all successful, indicating that the PN model is properly configured and capable of simulating the process without errors. The movement of tokens across the network enables the visualization of how objectives (goals) and sub-objectives (sub goals) are achieved by following the requirements and domain properties defined in the model.

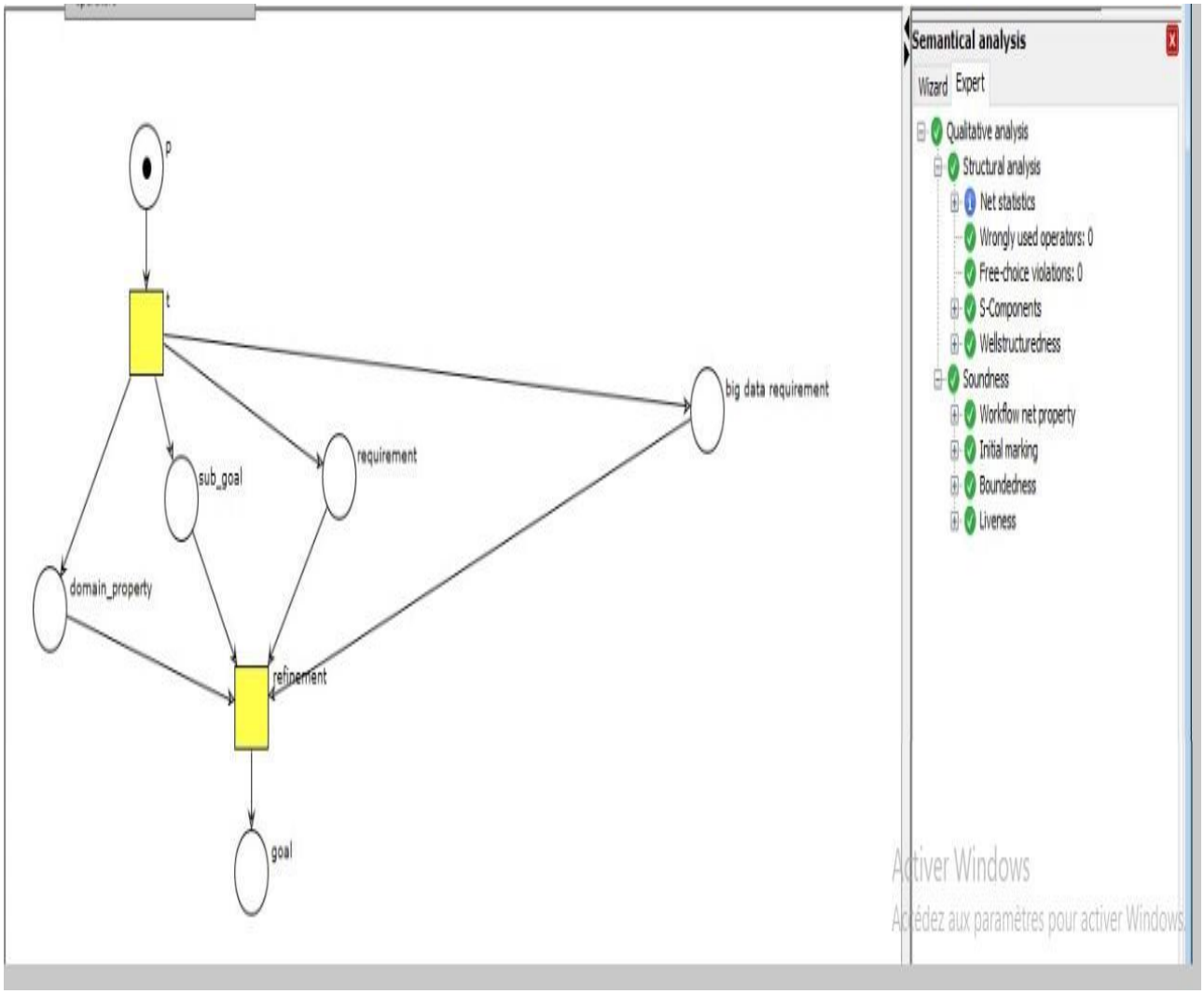

Figure 4.1: Analysis of a Petri net for the general case of BKAOS goal model

4.2.3.2 Responsibility model:

Structural Analysis:

Net Statistics:

Wrongly Used Operators: 0

Free-choice Violations: 0

S-Components: Wellstructuredness is confirmed.

Soundness:

Workflow Net Property: The net adheres to the workflow net property.

Initial Marking: The initial marking is correct.

Boundedness: The net is bounded.

Liveness: The net is live.

illustrates the PN simulation, which demonstrates the management of responsibilities and requirements to achieve an end goal. The movement of tokens across the network allows for the visualization of the process's progress. Semantic analysis confirms that the model is well structured, error-free, and can achieve the end goal consistently.

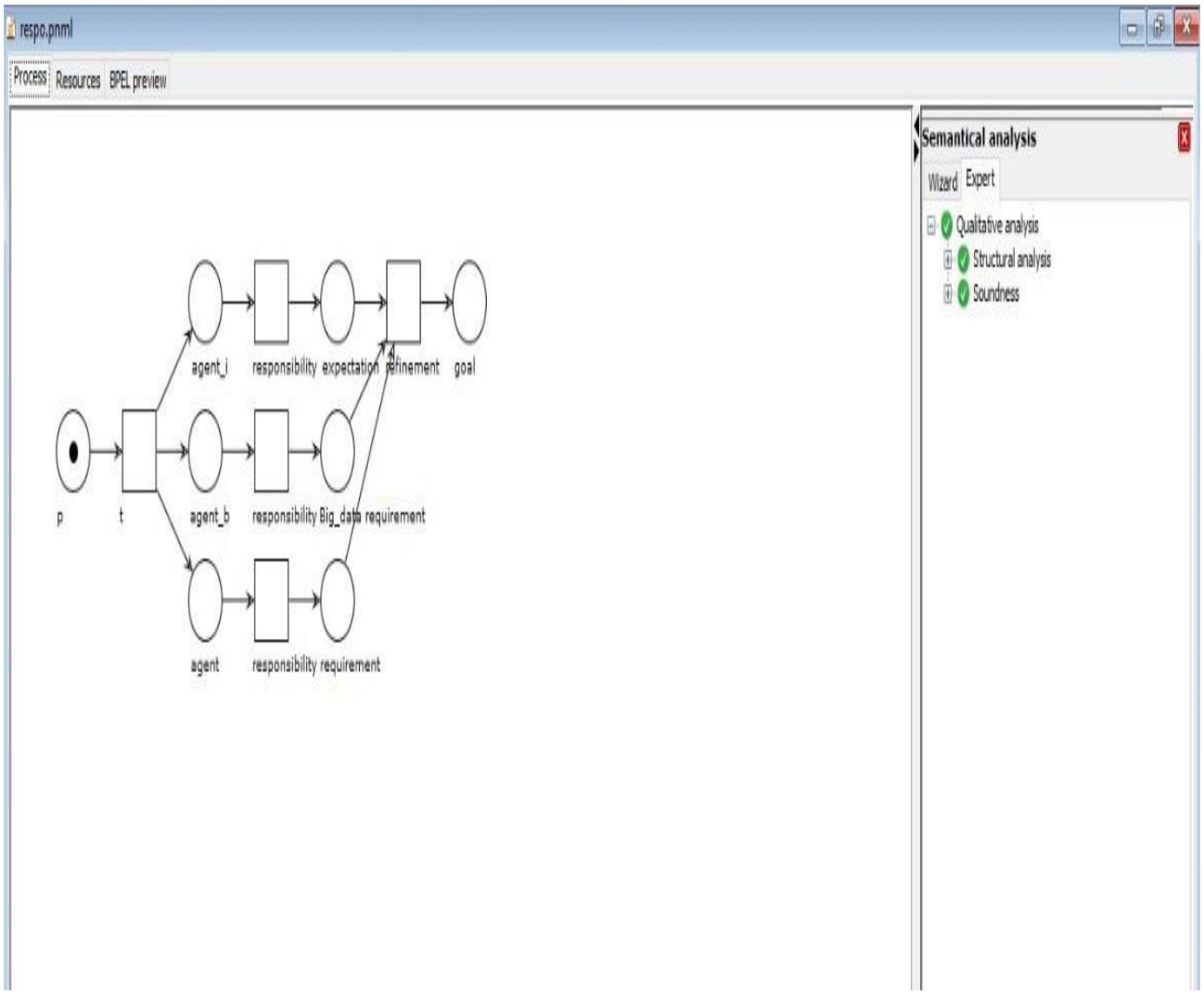

Figure 4.2: Analysis of a PN for the general case of BKAOS responsibility model

4.2.4 object model:

Structural Analysis:

Net Statistics:

Wrongly Used Operators: 0

Free-choice Violations: 0

S-Components: Wellstructuredness is confirmed.

Soundness:

Workflow Net Property: The net adheres to the workflow net property.

Initial Marking: The initial marking is correct.

Boundedness: The net is bounded.

Liveness: The net is live.

The workflow net depicted in the accompanying illustration is both structurally sound and functionally correct. The system has no errors in operator usage, adheres to the properties of the workflow net, and meets the criteria for boundedness and liveness, thereby ensuring reliable and continuous operation.

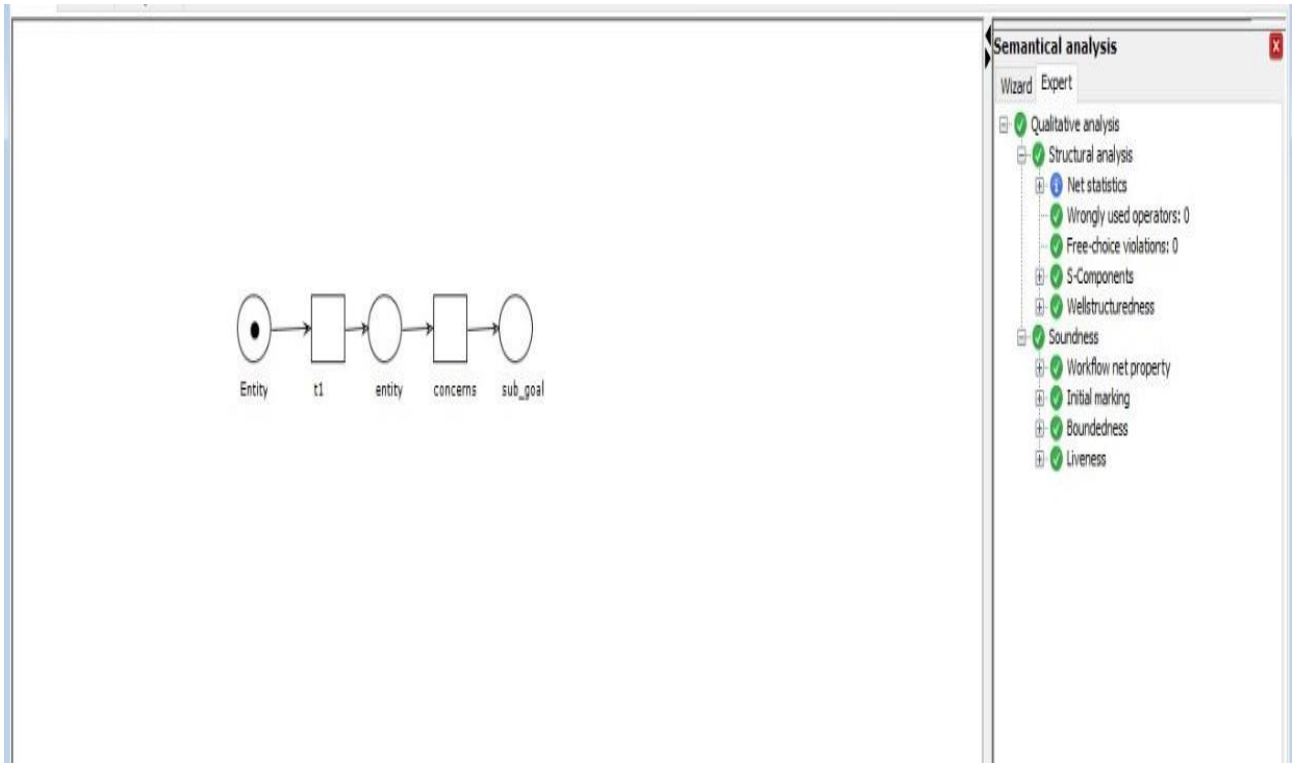

Figure 4.3: Analysis of a Petri net for the general case of BKAOS object model

4.2.5 operational model:

Structural Analysis:

Net Statistics:

Wrongly Used Operators: 0

Free-choice Violations: 0

S-Components: Wellstructuredness is confirmed.

Soundness:

Workflow Net Property: The net adheres to the workflow net property.

Initial Marking: The initial marking is correct.

Boundedness: The net is bounded.

Liveness: The net is live.

The simulation depicted in the figure represents a well-structured and sound workflow model. The analysis confirms that the model is free from structural errors and adheres to the principles of boundedness and liveness. This means that the workflow can proceed without deadlocks and that every transition can eventually be fired, ensuring smooth execution from start to end. The model effectively manages parallel activities and dependencies, making it a robust representation of a business process.

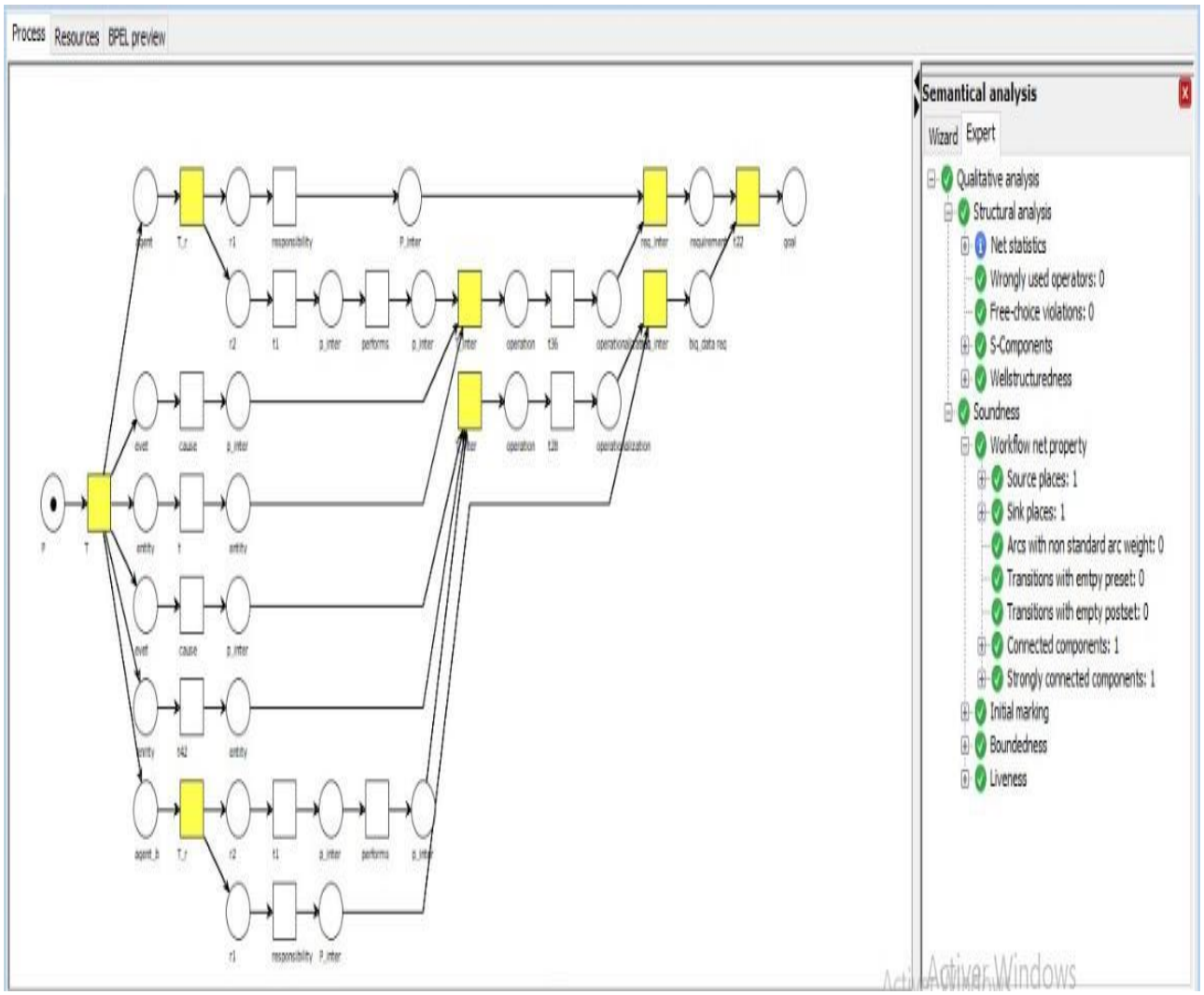

Figure 4.4: Analysis of a Petri net for the general case of BKAOS operational model

4.2.6 Analysis of a PN for the general case of BKAOS:

Structural Analysis:

Net Statistics:

Wrongly Used Operators: 0

Free-choice Violations: 0

S-Components: Wellstructuredness is confirmed.

Soundness:

Workflow Net Property: The net adheres to the workflow net property.

Initial Marking: The initial marking is correct.

Boundedness: The net is bounded.

Liveness: The net is live.

The results demonstrate that the complex PN is both structurally and functionally correct. It is well-structured, with no errors in operator usage or free-choice violations. It adheres to the workflow net properties, has valid initial markings, and is bounded. All transitions are live, meaning the process can proceed without deadlocks, ensuring a smooth and reliable workflow.

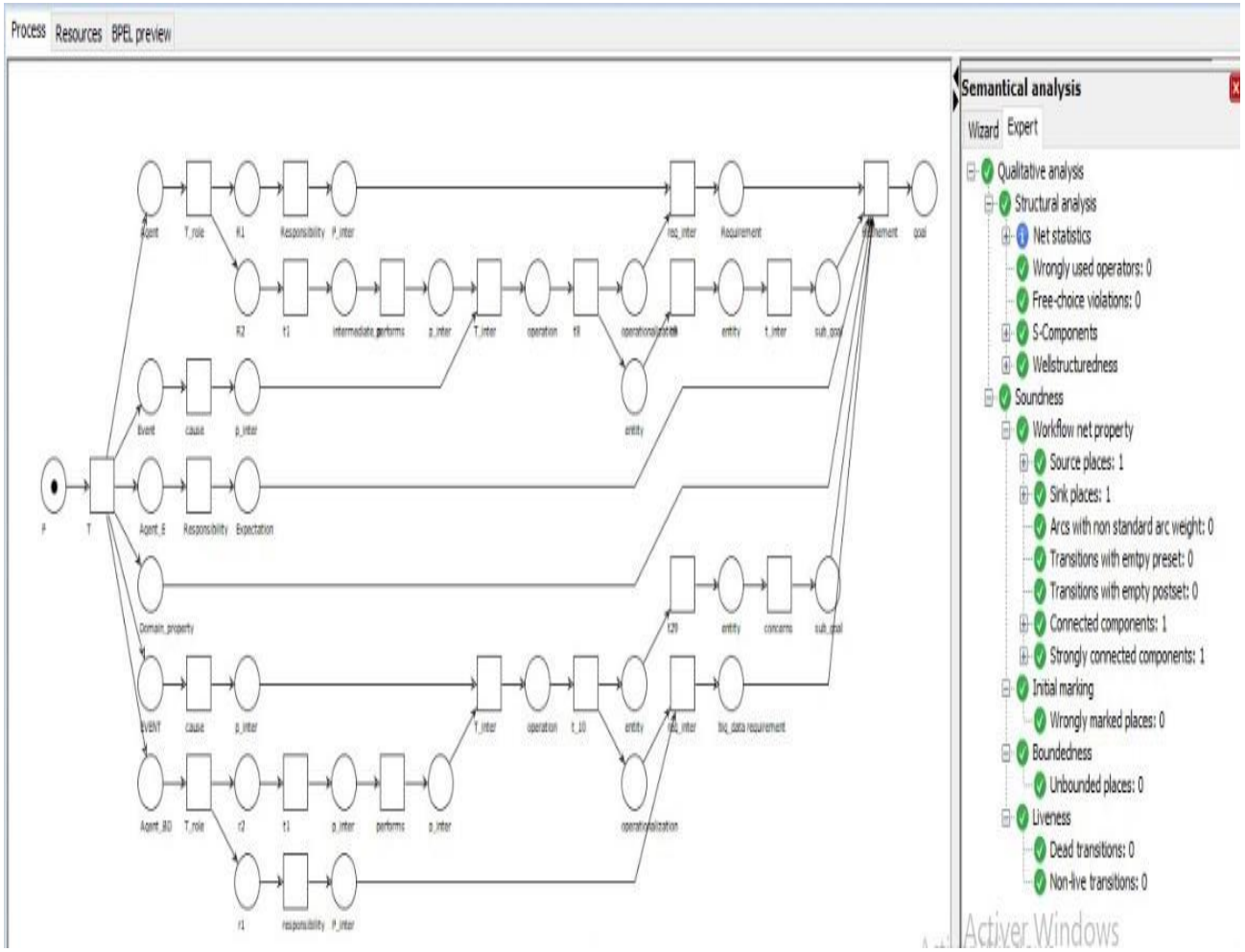

Figure 4.5: Analysis of a PN for the general case of BKAOS meta model

4.3 Bigraphs Education :

4.3.1 Application of the Bigraphs in the general case of KAOS:

The KAOS semantics is defined by a Bigraph
$Bi{-}KAOS = V\,KAOS, EKAOS, CtrlKAOS, G^{P}\,KAOS, G^{L}KAOS : IKAOS{-})JKAOS$,
where:

- **Set of nodes:**

$$V\,KAOS = \{v_{goal}, v_{subgoal}, v_{Agent_i}, v_{Agent_j}, v_{Requirement}, v_{Operation}, v_{Expectation}, v_{DomainProperty}, v_{Entity_i}, v_{Entity_j}, v_{Event}\}$$

- **Set of edges:**

$$EKAOS = \{e_{Refinement}, e_{Resporeq}, e_{Respoexp}, e_{Performs}, e_{Operationalisation}, e_{Conserns}, e_{Entity}, e_{cause}\}$$

$e_{Refinement}$ l'hyper-arc that relies the requierements, expectations, entity, subgoal and the domain proprety to the goal.

$e_{Resporeq}$ l'hyper-arc that relies v_{Agent_i} with the $v_{Requirement}$ defining the responsibility of the agent on requirements .

$e_{Respoexp}$ l'hyper-arc that relies v_{Agent_j} with the $v_{Expectation}$ defining the responsibility of the agent on expectation.

$e_{performs}$ l'hyper-arc that relies an v_{Agent_i} to the $v_{Operation}$ he should perform.

$e_{Operationalisation}$ l'hyper-arcs that relies $v_{Operation}$ to $v_{Requirement}$.

$e_{Conserns}$ l'hyper-arc that relies v_{Entity_i} to $v_{subgoal}$

e_{Entity} l'hyper-arc that relies an entity to an other.

e_{cause} l'hyper-arc that relies v_{Event} to $v_{Operation}$

- **Set of controls for each node:**

  $CtrlKAOS(v_{Goal}) = atomic : 4,$

  $CtrlKAOS(v_{Agent_i}) = atomic : 2,$

  $CtrlKAOS(v_{Agent_j}) = atomic : 1,$

  $CtrlKAOS(v_{Requirement}) = atomic : 3,$

  $CtrlKAOS(v_{Operation}) = atomic : 4,$

  $CtrlKAOS(v_{Expectation}) = atomic : 2,$

  $CtrlKAOS(v_{DomainProperty}) = atomic : 1,$

  $CtrlKAOS(v_{Entity_i}) = atomic : 2,$

  $CtrlKAOS(v_{Entity_j}) = atomic : 1,$

  $CtrlKAOS(v_{Event}) = atomic : 1,$

- G^P *KAOS* is the place graph that particularly represents the parent function defined as: prnt: siteo U VKAOS -) VKAOS U regiono, knowing that: prnt(v_{Goal}) = prnt(v_{Agent_i}) = prnt(v_{Agent_j}) = prnt($v_{Requirement}$) = prnt($v_{Operation}$) = prnt($v_{Expectation}$) =prnt($v_{DomainProperty}$) =prnt(v_{Entity_i}) = prnt(v_{Entity_j}) =prnt(v_{Event}) =regiono .

Figure 4.6 shows the application of the place graph on KAOS.

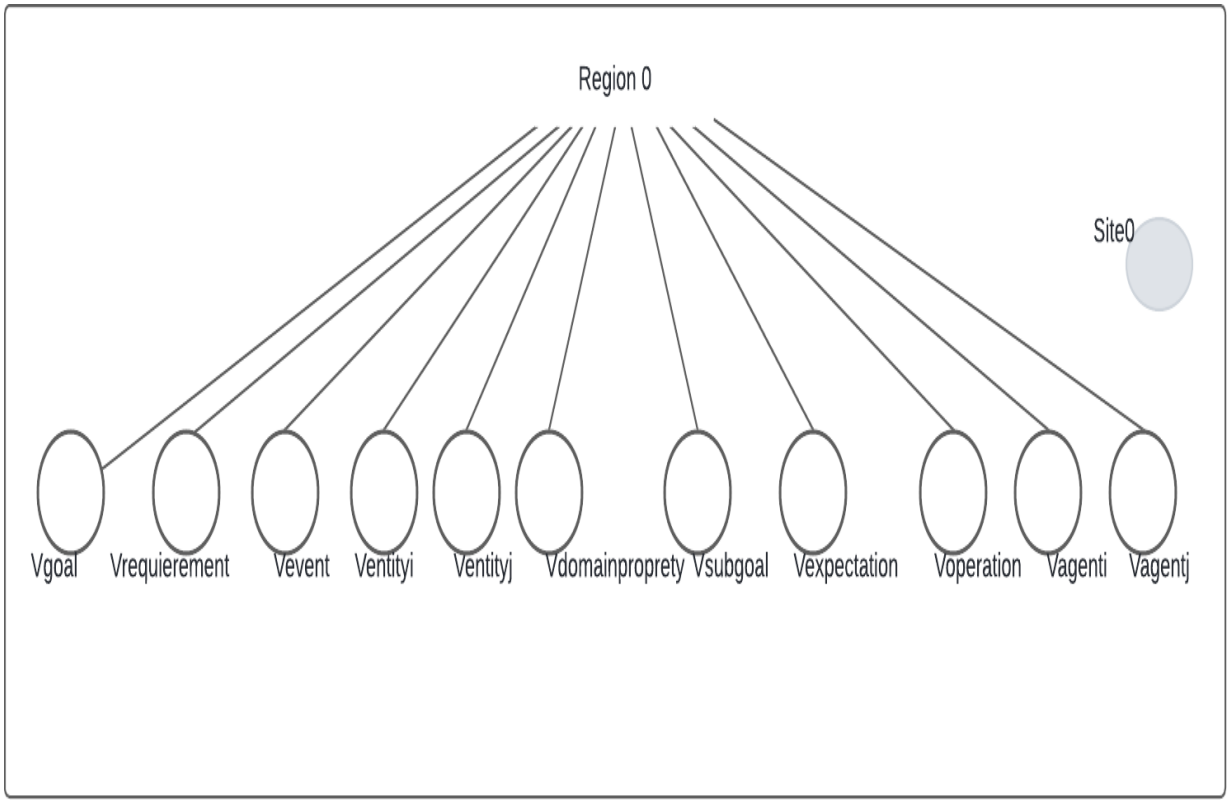

Figure 4.6: Place graphe for Bi-KAOS

- $G^{L}KAOS$ is the graph of links that particularly represent the link function defined as:

  link :ϕUP)$EKAOSU\phi$, P is the set of ports p11, p12, etc..,

  link(p011) =$e_{domfinement}$, link(p1) =$e_{reqfinement}$, link(p4) =$e_{Expfinement}$, link(p6) =$e_{subfinement}$ link(p10) = $e_{Operationalisation}$, link(p20) = $e_{Resporeq}$, link(p30)= $e_{Performs}$, link(p40)=e_{Entity} , link(p50)= e_{Cause}, link(p60)=$e_{Concerns}$, link(p70)= $e_{Respoexpt}$.

  IKAOS = $(1, \phi)$, without inner names and having one site that abstracts the possible insertion of other nodes,

  JKAOS = $(1, \phi)$, without outer names and having one region

Figure 4.7 graphically shows the application of Bigraph on the KAOS model; there is one region, a site, the nodes (v_{goal}, $v_{subgoal}$, v_{Agent_i}, v_{Agent_j}, $v_{Requirement}$, $v_{Operation}$, $v_{Expectation}$, $v_{DomainProperty}$, v_{Entity_i}, v_{Entity_j}, v_{Event}}), their ports, and the relationships between them.

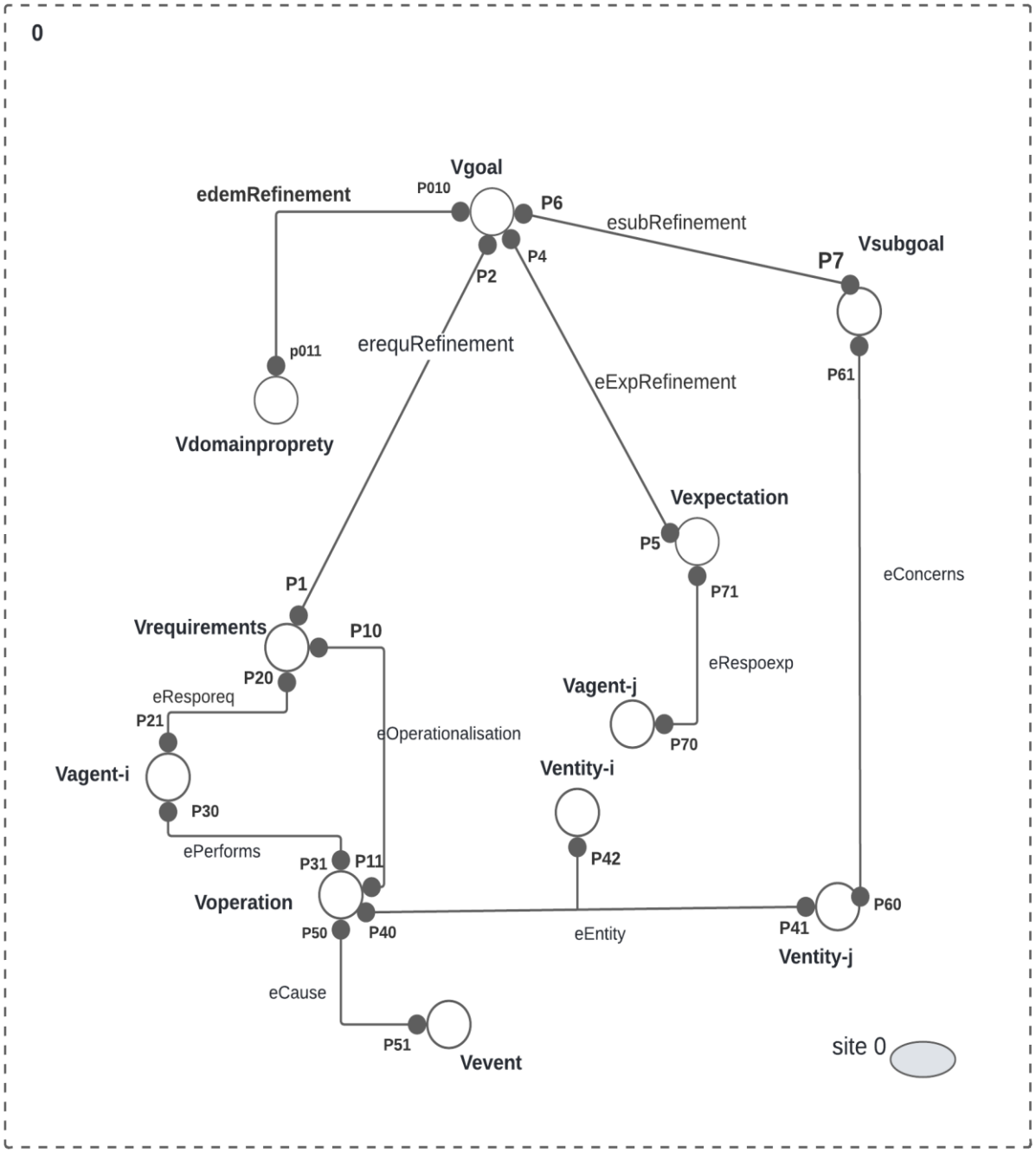

Figure 4.7: Bi-KAOS a Bigraphs for the general case of KAOS meta model

4.3.2 Application of the Bigraphs in the general case of BKAOS:

The BKAOS semantics are defined with the KAOS semantics withing the big data characteristics (big data requirements)

$Bi - KAOS = V\,BKAOS, EBKAOS, CtrlBKAOS, G^P\,BKAOS, G^L BKAOS : IBKAOS-)JBKAOS$,with

- **Set of nodes for Bi-BKAOS** same with the KAOS nodes plus the bigdatarequirements node and the nodes related with the operation committed on bigdatarequirements

$V\,BKAOS = \{v_{goal}, v_{subgoal}, v_{Agent_i}, v_{Agent_j}, v_{Requirements}, v_{Operation}, v_{Expectation}, v_{DomainProperty}, v_{Entity_i}, v_{Entity_j}, v_{Event}, v_{BDRequirements}, v_{BDsubgoal}, v_{\upsilon_{BAgent_i}}, v_{BDOperation}, v_{BDEntity_i}, v_{BDEntity_j}, v_{BDEvent}\}$

- **Set of edges for Bi-BKAOS:**

$EBKAOS = \{e_{Refinement}, e_{Resporeq}, e_{Respoexp}, e_{Performs}, e_{Operationalisation}, e_{Conserns}, e_{Entity}, e_{cause}, e_{RespoBDreq}, e_{BDPerforms}, e_{BEntity}, e_{Bcause}, e_{BConserns}\}$

same edges as Bi-KAOS with adding this :

$e_{Refinement}$ l'hyper-arc that relies the requirements,the big data requirements, expectations, subgoal ,bigDatasubgoal and the domain proprety to the goal.

$e_{BDPerforms}$arc that relies an v_{BAgent_i} to the $v_{BDOperation}$ he should perform.

$e_{RespoBDreq}$arc that relies v_{BAgent_i} with the $v_{BDRequirement}$ defining the responsibility of the agent on requirements .

$e_{BEntity}$to relie BDentities

e_{Bcause}arc that relies $v_{BDEvent}$ to $v_{BDOperation}$

$e_{BDOperationalis}$arc that relies $v_{BDOperation}$ to $v_{BDRequirement}$.

$e_{Bconcerns}$arc that relies $v_{BDEntity_i}$ to $v_{BDsubgoal}$

- **Set of controls for each node:**

$CtrlBKAOS(v_{Goal}) = atomic : 4,$

$CtrlBKAOS(v_{Subgoal}) = atomic : 2,$

$CtrlBKAOS(v_{BDsubgoal}) = atomic : 2,$

$CtrlBKAOS(v_{Agent_i}) = atomic : 2,$

$CtrlBKAOS(v_{BAgent_i}) = atomic : 2,$

$CtrlBKAOS(v_{Agent_j}) = atomic : 1,$

$CtrlBKAOS(v_{Requirement}) = atomic : 3,$

$CtrlBKAOS(v_{BDRequirement}) = atomic : 3,$

$CtrlBKAOS(v_{Operation}) = atomic : 4,$

$CtrlBKAOS(v_{BDOperation}) = atomic : 4,$

$CtrlBKAOS(v_{Expectation}) = atomic : 2,$

$CtrlBKAOS(v_{DomainProperty}) = atomic : 1,$

$CtrlBKAOS(v_{Entity_i}) = atomic : 2,$

$CtrlBKAOS(v_{BDEntity_i}) = atomic : 2,$

$CtrlBKAOS(v_{Entity_j}) = atomic : 1,$

$CtrlBKAOS(v_{BDEntity_j}) = atomic : 1,$

$CtrlBKAOS(v_{Event}) = atomic : 1,$

$CtrlBKAOS(v_{BDEvent}) = atomic : 1,$

- $G^P BKAOS$ is the place graph that particularly represents the parent function defined as: prnt: site0 U VBKAOS -) VBKAOS U region0,and site1 U VBKAOS -)VBKAOS u region1 knowing that:
  - (v_{Goal}) = prnt(v_{Agent_i}) = prnt(v_{Agent_j}) = prnt($v_{Requirement}$) = prnt($v_{Operation}$) = prnt($v_{Expectation}$) =prnt($v_{DomainProperty}$) =prnt(v_{Entity_i}) = prnt(v_{Entity_j}) =prnt(v_{Event}) =region0 .
  - prnt(v_{BAgent_i}) = prnt($v_{BDRequirement}$) = prnt($v_{BDOperation}$) =prnt($v_{DomainProperty}$) =prnt($v_{BEntity_i}$) = prnt($v_{BEntity_j}$) =prnt(v_{BEvent}) =region1 .

.

Figure 4.8 shows the application of the place graph on BKAOS.

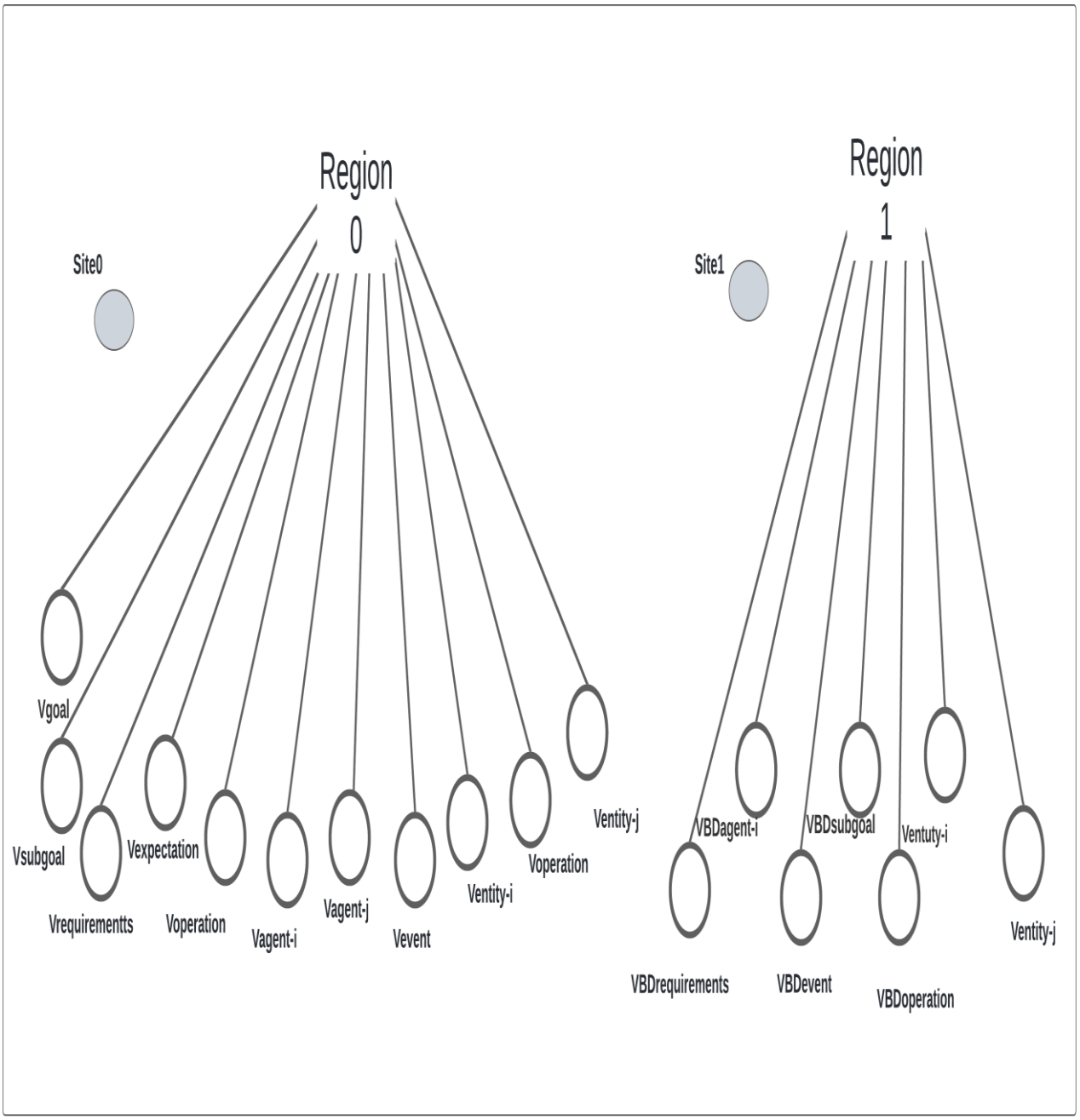

Figure 4.8: $G^P BKAOS$ Place graph for BKAOS

- $G^L BKAOS$ is the graph of links that particularly represent the link function defined as:

  link :$\phi UP)EKAOSU\phi$, P is the set of ports p11, p12, etc..,

  link(p011) =$e_{domRefinement}$,link(p1) =$e_{reqRefinement}$, link(p5) =$e_{expRefinement}$, link(p6) =$e_{subGRefinement}$, , link(p10) = $e_{Operationalisation}$,

link(p20) = $e_{Resporeq}$, link(p30)= $e_{Performs}$, link(p40)=e_{Entity} , link(p50)= e_{Cause}, link(p60)=$e_{Concerns}$, link(p70)= $e_{Respoexpt}$, link(p80) =$e_{respoBDreq}$, link(p90) =$e_{eBDperforms}$, link(p100) =$e_{BDoperationalisation}$,link(p110) =$e_{BDentity}$, link(p120) =$e_{BDcause}$, link(p130) =$e_{BDentity}$,link(p140) =$e_{BDconcerns}$,

IBKAOS = $(2, \phi)$, without inner names and having tow site that abstracts the possible insertion of other nodes,

JBKAOS = $(2, \phi)$, without outer names and having tow regions

Figure 4.9 graphically shows the link graph of the BKAOS model;

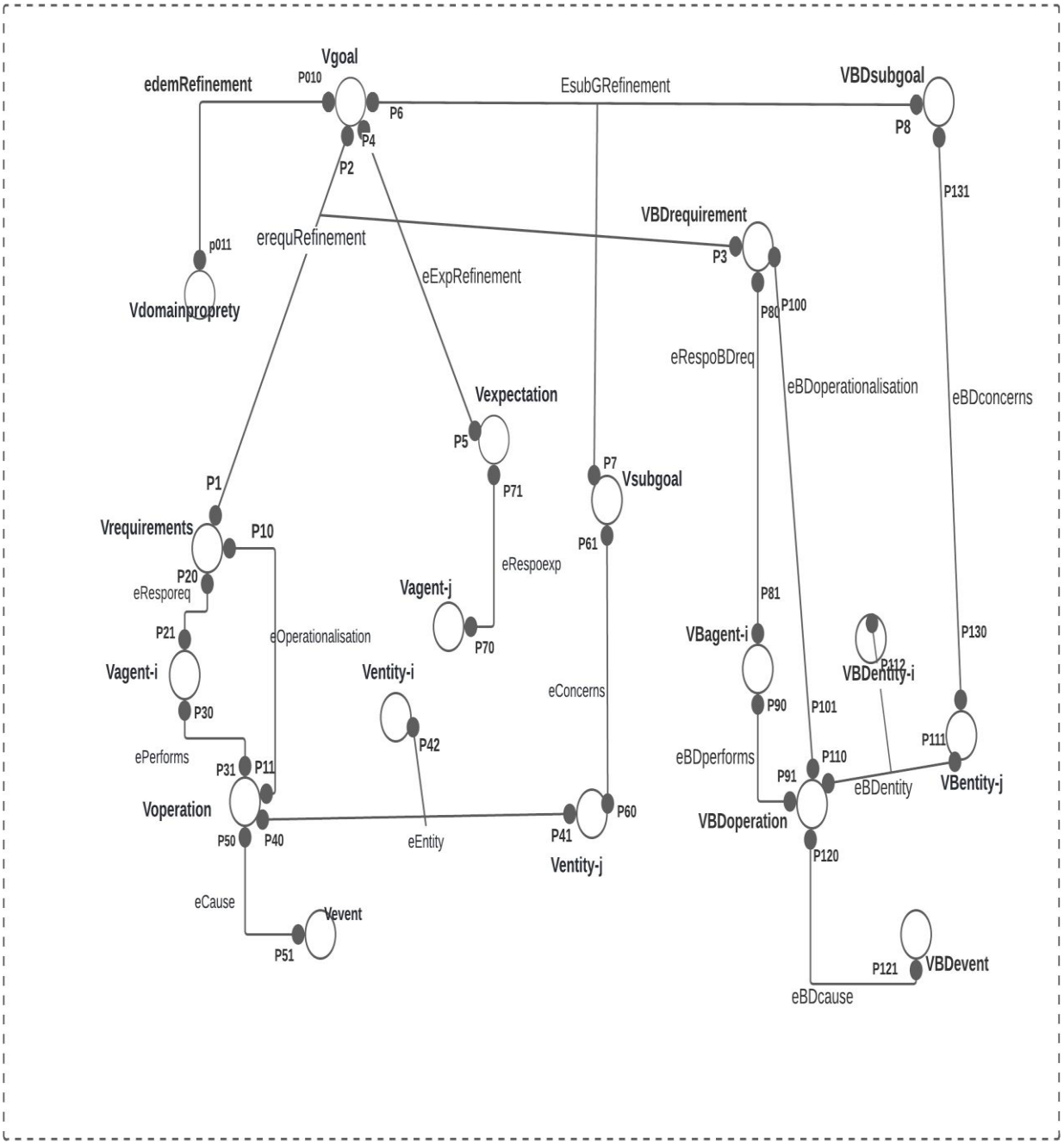

Figure 4.9: *G^LBKAOS* graph of links Bigraphs

Figure 4.10 graphically shows the application of Bigraph on the BKAOS model; there is two regions(region0, region1) , two sites(site0,site1),
the nodes (v_{goal}, $v_{subgoal}$, v_{Agent_i}, v_{Agent_j}, $v_{Requirement}$, $v_{Operation}$, $v_{Expectation}$, $v_{DomainProperty}$, v_{Entity_i}, v_{Entity_j}, v_{Event}, $v_{BDRequirement}$, , $v_{BDsubgoal}$, $v_{\upsilon_{BAgent_i}}$, $v_{BDOperation}$, $v_{BDEntity_i}$, $v_{BDEntity_j}$, $v_{BDEvent}$), their ports, and the relationships between them.

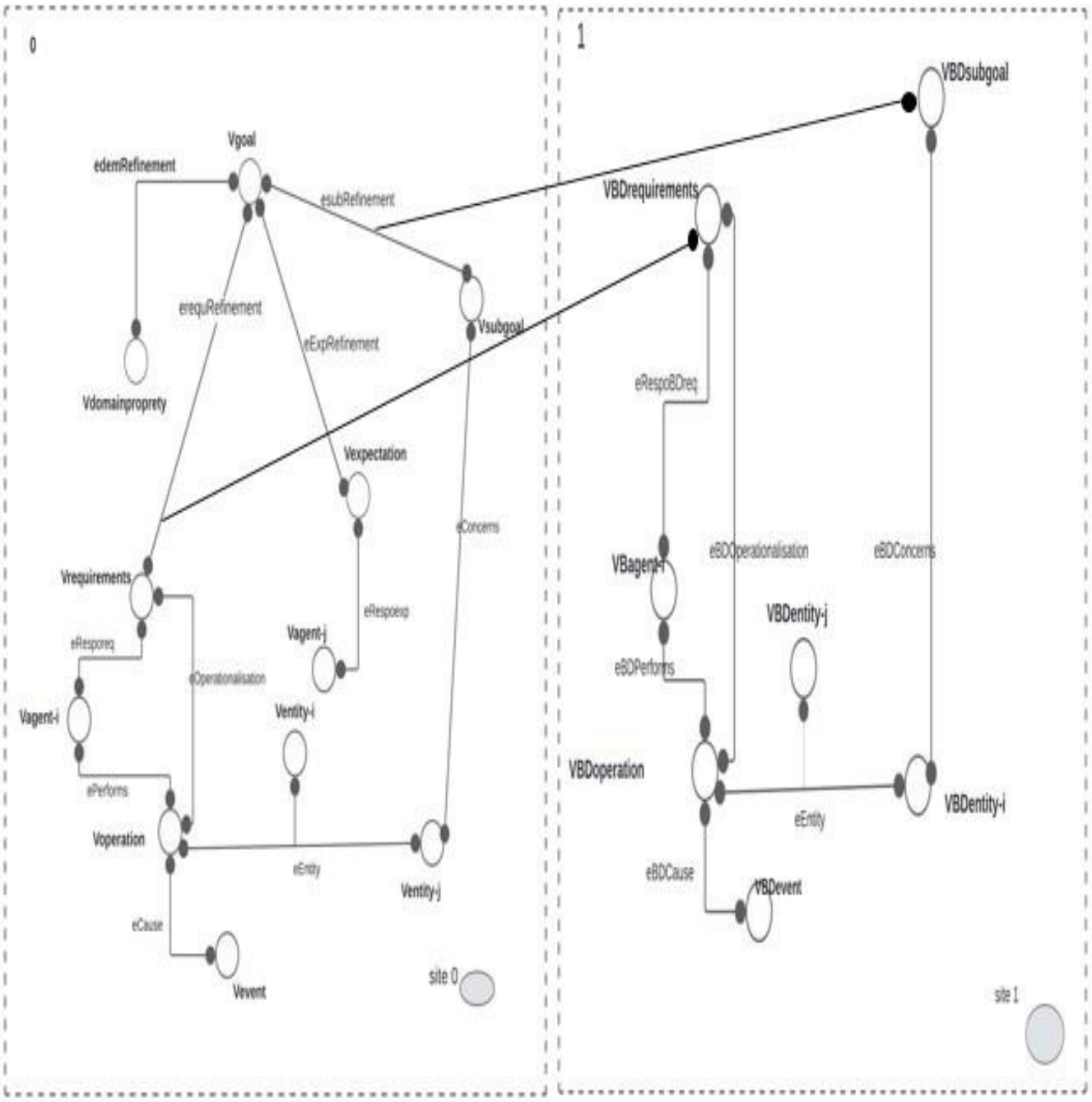

Figure 4.10: Bi-BKAOS a Bigraphs for the general case of BKAOS meta model

4.3.3 Conclusion:

The formal verification of BKAOS was conducted in this chapter using two models: Petri nets and bigraphs. By using these models, it was possible to conduct a thorough analysis and simulation of the system's behavior in different scenarios. Petri nets verification revealed potential deadlocks, bottlenecks, and made sure the system's execution sequences were correct. Meanwhile, bigraphs verification emphasized the locality and con-

nectivity aspects of the BKAOS method.The integration of Petri nets and bigraphs not only enhanced our understanding of the BKAOS method but also confirmed its reliability and correctness..

As Bigraph provides and petri nets strong mathematical basis, another perspective is to build a framework that allows for the validation of the integrated property of any KAOS extension..

General conclusion and perspectives

Previously requirements engineering did not effectively support the elicitation process for big data projects. Essential characteristics of big data, such as (volume, variety ,.....), were not adequately considered in traditional requirements engineering frameworks. This gap led researchers to develop new methodologies to address these unique challenges.

One notable advancement is BKAOS, an extension of the KAOS (Knowledge Acquisition in Automated Specification) methodology. The BKAOS method is specifically designed to elicit and manage the requirements of big data projects, integrating considerations for the distinctive attributes of big data.

In the course of our investigation, we subjected the BKAOS to a rigorous examination through the application of two sophisticated analytical models. The models employed were those of Petri nets and bigraphs. Petri nets, which are widely recognized for their robustness in modeling concurrent systems, allowed us to visually and mathematically capture the various states, transitions, and interactions within BKAOS. This model is particularly effective in identifying potential deadlocks, bottlenecks, and ensuring the system's correct execution sequences.

The use of Bigraphs, which combine elements of graph theory and mobile processes, provided a complementary perspective by representing both the spatial configuration and the dynamic connectivity of the system. This dual capability enabled us to model not just the static structure of BKAOS, but also its dynamic evolution over time, capturing changes in both the physical and logical relationships within the system.

Our verification process included a series of rigorous testing scenarios, which simulated various operational conditions to observe and measure the system's behavior. The results of these simulations were extremely positive, demonstrating the robustness and reliability of BKAOS.

Perspective.1 As we conclude our discussion on our current contributions, it's important to highlight another perspective for future work. Given the robust mathematical foundation provided by Bigraphs and Petri nets, we propose to develop new framework that builds upon existing findings. This framework will focus on automating the generation of Petri nets or Bigraphs graphs for any scenario provided by users. Using artificial intelligence, it'll translate concepts from the BKAOS model, such as requirements and goals, into the corresponding elements of Petri nets (like places and transitions) and Bigraphs (such as nodes, arities, and signatures). Ultimately, it'll simulate the example and validate it to ensure its effectiveness.

Perspective.2 Since BKAOS stands out as one of the few extensions tailored to meet the requirements of Big Data projects during the elicitation phase and has been verified as an effective method for Big Data requirements elicitation, we propose extending this approach further. By applying BKAOS to numerous projects and obtaining validation, we intend to enhance its applicability. Additionally, we suggest developing extensions for other methods to address the unique characteristics of Big Data and performing formal verification on these new extended methods to ensure their integrity.

Bibliography

[1] https://dotnettutorials.net/lesson/5-vs-of-big-data/ . 21/04/2024.

[2] Respect-IT. A kaos tutorial ,v1.0. Oct. 18, 2007.

[3] Chris Hersman and Kim Fowler. *Best practices in spacecraft development*. Elsevier, 2010.

[4] Mohd Sadiq and SK Jain. An insight into requirements engineering processes. pages 313–318, 2012.

[5] https://www-geeksforgeeks-org.cdn.ampproject.org . 15/02/2024.

[6] Abbas Ahmad Kamran Khalid Muhammad Wasim Pardeep Kumar, Aisha khan. Software engineering requirement model. November-2018.

[7] *the lesson of requirements engineering university cape of town south afrika*.

[8] Nagy Ramadan and Salwa Megahed. Requirements engineering in scrum framework. *International Journal of Computer Applications*, 149:24–29, 09 2016.

[9] Alistair Sutcliffe. Scenario-based requirements engineering. in proceedings of the 11th ieee international conference on requirements engineering, pages 320–, washington, dc, usa. 2003.

[10] Colette Rolland, Carine Souveyet, and Camille Ben Achour. Guiding goal modeling using scenarios. *IEEE transactions on software engineering*, 24(12):1055–1071, 1998.

[11] Axel van Lamsweerde. Requirements engineering : From system goals to uml models to software specifications. *Wiley*, 2009.

[12] Awais Rashid et Ruzanna Chitchyan. requirements engineering : a roadmap. in proceedings of the 13th international workshop on early aspects. 2008.

[13] Michael Jackson. Problem frames. "analyzing and structuring software development problems". *Addison-Wesley Longman Publishing Co*, 2001.

[14] Evellin Cardoso Tong Li Alejandro Maté Elda Paja† Mattia Salnitri John Mylopoulos† Paolo Giorgini Jennifer Horkoff, Fatma Ba¸sak Aydemir. "goal-oriented requirements engineering: A systematic literature map".

[15] A. van Lamsweerde. "requirements engineering in the year 00: A research perspective". *ACM Press*, 2000.

[16] Nisar Hundewale Sultan Aljahdali, Jameela Bano. "goal oriented requirements engineering - a review".

[17] Abderrahman MATOUSSI. Thése: Construction de spécifications formelles abstraites dirigée par les buts. décembre 2011.

[18] Yixuan Hou Yunduo Wang Yiwen Chen, Yuanpeng Wang. "t-star: A text-based istar modeling tool". *Beijing University of Technology Beijing, China*, 2019.

[19] Enyo Gonçalves· Marcos Antônio de Oliveira · Ingrid Monteiro · Jaelson Castro · João Araújo. "understanding what is important in istar extension proposals: the viewpoint of researchers". *Springer-Verlag London Ltd., part of Springer Nature*, 2018.

[20] Hisayuki Horai et Motoshi Saeki Haruhiko Kaiya. Agora : Attributed goal-oriented requirements analysis method. 2002.

[21] Kridanto Surendro Sandfreni. Requirements engineering model: Role based goal oriented model. 2016.

[22] Manjunath Kamath Ishita Gupta. Adding value to manufacturing, retail, supply chain, and logistics operations with big data analytics. *School of Industrial Engineering and Management Oklahoma State University*.

[23] Imed Riadh Farah Imen Chebbi, Wadii Boulila. Big data: Concepts, challenges and applications. *Laboratoire RIADI, Ecole Nationale des Sciences de L'Informatique, Manouba, Tunisia*, September 2015.

[24] http://www.gartner.com/it-glossary/bigdata/ . 20/04/2024.

[25] BOURAHDOUN MOHAMMED ILYAS. Memoire : Impact des méthodes analytiques dans le contexte des données massives. *Université de 8 Mai 1945 – Guelma* -, Octobre 2020.

[26] Rhythm Bhatia Manpreet Singh, Vandan Bhatia. Big data analytics solution to healthcare. *Manipal University Jaipur*, Dec 22-23, 2017.

[27] https://www.integrate.io/blog/structured-vs-unstructured-data-key-differences/ . 20/04/2024.

[28] A. M. CAMACHO D. MERAYO, A. RODRÍGUEZ-PRIETO. Prediction of physical and mechanical properties for metallic materials selection using big data and artificial neural networks. *Digital Object Identifier 10.1109/ACCESS.2020.2965769*, January 22, 2020.

[29] Anca APOSTU Manole VELICANU Elena Geanina ULARU, Florina Camelia PUICAN. Perspectives on big data and big data analytics. *Database Systems Journal vol. III*, Database Systems Journal vol. III, no. 4/2012.

[30] Aurélie Dudezert Myriam Karoui, Grégoire Davauchelle. Big data : Mise en perspective et enjeux pour les entreprises. *ngénierie des systèmes d information*, June 2014.

[31] Pelánek. Formal verification model checking. 2020.

[32] 15/02/2024. https://opencourses.emu.edu.tr/pluginfile.php/53492/mod_resource/content/3/Formal%20introduction%20to%20Petri%20nets.pdf.

[33] Slim BENSAOUD. Cours.

[34] Etienne Renault. Réseaux de petri. Avril 2015.

[35] https://teaching.model.in.tum.de/2021ss/petri/material/petrinets.pdf#:~:text=URL%3A%20https%3A%2F%2Fteaching.model.in.tum.de%2F2021ss%2Fpetri%2Fmaterial%2Fpetrinets.pdf%0AVisible%3A%200%25%20.

[36] Chapitre 4 les reseaux de petri. (n.d.) https://www.uvt.rnu.tn/resources-uvt/cours/automatismes/chapitre4_rdp.pdf.

[37] Marc Bourcerie. Master recherche systèmes dynamiques et signaux la modelisation des systemes de production par reseaux de petri. 2010-2011.

[38] Robin Milner. The space and motion of comunicating agents. Decembre 1, 2008.

[39] Chabane Djeddi, Nacer Eddine Zarour, and Pierre-Jean Charrel. Formal verification of extension of istar to support big data projects. *Computer Science*, 22, 2021.

[40] Taha Abdelmoutaleb Cherfia. *Thése : Un Framework Basé Bigraphes pour la Conception et l'Analyse des Systèmes Sensibles au Contexte*. Constantine, Algérie, 2016.

[41] Ebbe Elsborg, Thomas T Hildebrandt, and Davide Sangiorgi. Type systems for bigraphs. In *Trustworthy Global Computing: 4th International Symposium, TGC 2008, Barcelona, Spain, November 3-4, 2008, Revised Selected Papers 4*, pages 126–140. Springer, 2009.

[42] Z Benzadri. *Spécification et Vérification Formelle des Systemes Cloud*. PhD thesis, Thesis, Université Constantine 2-Abdelhamid Mehri, 2016.

[43] Ayoub Bouheroum, Zakaria Benzadri, and Faiza Belala. Towards a formal approach based on bigraphs for fog security: Case of oil and gas refinery plant. In *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 64–71. IEEE, 2019.

[44] Nadira Benlahrache, Faiza Belala, Chafia Bouanaka, and Malika Benammar. Vers un modèle de déploiement à base de bigraphes. In *CAL*, pages 131–143, 2010.

[45] Julio Cesar Sampaio do Prado Leite Vera Maria Bejamim Werneck, Antonio de Padua Albuquerque Oliveira. Comparing gore frameworks: i-star and kaos. *University of the State of Rio de Janeiro ,Department of Informatics, Rua Marques de São Vicente 225*, RJ, Brazil.

[46] Nacer eddine Zarour 1 Chabane Djeddi 1 and Pierre-Jean Charrel 2. A requirement elicitation method for big data projects. *LIRE Laboratory, Constantine 2- A. Mehri University, Algeria ,IRIT Laboratory, Toulouse 2 Jean Jaurès University, France*.

[47] Abbas Ahmad Kamran Khalid Muhammad Wasim Pardeep Kumar, Aisha khan. Software engineering requirement model. 2009.

[48] Blair Archibald, Muffy Calder, Michele Sevegnani, and Mengwei Xu. Modelling and verifying bdi agents with bigraphs. *Science of Computer Programming*, 215:102760, 2022.

[49] Lian Yu, Wei Tek Tsai, Yanbing Jiang, and Jerry Gao. Generating test cases for context-aware applications using bigraphs. In *2014 Eighth International Conference on Software Security and Reliability (SERE)*, pages 137–146. IEEE, 2014.

[50] Camille Salinesi. *L'Ingénierie des Exigences appliquée aux Systèmes d'Information*. PhD thesis, Université Panthéon-Sorbonne-Paris I, 2010.

[51] Osman Hasan and Sofiene Tahar. Formal verification methods. In *Encyclopedia of Information Science and Technology, Third Edition*, pages 7162–7170. IGI global, 2015.

[52] Sultan Aljahdali, Jameela Bano, and Nisar Hundewale. Goal oriented requirements engineering-a review. In *24th International Conference on Computer Applications in Industry and Engineering, Honolulu, Hawaii, USA, CAINE*, pages 16–18, 2011.

[53] Shuichiro Yamamoto, Haruhiko Kaiya, Karl Cox, and Steven Bleistein. Goal oriented requirements engineering: trends and issues. *IEICE TRANSACTIONS on Information and Systems*, 89(11):2701–2711, 2006.