Indonesian Journal of Computer Science
ISSN 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam No. 1, Padang, Indonesia
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Goal-Oriented Modeling of an Urban Subway Control System Using KAOS

Lokanna Kadakolmath[1], Umesh D. Ramu[2]
lokanna@acharya.ac.in, drumesh@pesce.ac.in
[1]Department of Information Science & Engineering, Acharya Institute of Technology, Bengaluru 560 107, India
[2]Department of Computer Science & Engineering, P.E.S College of Engineering, Mandya 571 401, India

Article Information

Submitted : 14 Jun 2023
Reviewed: 21 Jun 2023
Accepted : 27 Jun 2023

Keywords

Goal Modeling; KAOS; Requirements Engineering; Semi-formal model; Safety-critical systems.

Abstract

The extent to which a safety-critical system, such as an urban subway control system, accomplishes its goals is a fundamental metric of its success. Identifying and assessing these goals should thus be one of the primary tasks in safety-critical system development. The breakdown of these systems may result in the loss of human lives and assets. The failure of these systems is caused by insufficient, incomplete, ambiguous, or conflicting requirements. Non-functional requirements are also separated from requirement specifications. Goal-oriented requirements engineering methodologies, such as KAOS, are used to tackle these challenges by providing adequate, complete, unambiguous, and consistent requirements in terms of goals. As a result, the KAOS approach is utilized in this article to construct a goal-oriented model of an urban subway control system. Variability and obstacle concerns are also addressed in this study.

A. Introduction

To ensure that the software being used successfully addresses a certain issue, analysts have to precisely comprehend and explain the problem. At first glance, this appears to be common sense. However, as we will see, determining the specific nature of the problem can be difficult. As a result, it is essential to identify, comprehend, communicate, investigate, and decide on which issues will be addressed, why such an issue must be solved, and who will be held accountable for fixing that issue. This is commonly referred to as requirements engineering (RE) [1], [2].

Many research investigations [3], [4] have demonstrated that a fault in the requirements engineering phase is the primary reason for failure in software development initiatives. Several incidents are presented on the website "Forum on Risks to the Public in Computers and Related Systems," exhibiting insufficient requirements analysis [5]. A requirement engineering flaw causes 40% of software projects to malfunction or not succeed in achieving most of the predicted requirements. Bell and Thayer said in 1976 that nonconformity of system functionalities to user needs, as well as the inadequacy, inconsistency, and vagueness of requirements documents, are all factors that lower software quality [6]. According to Bell and Thayer, "the requirements for a system do not arise naturally; rather, they must be engineered and subjected to ongoing review and revision."

The RE association has worked hard to improve the requirements engineering phase of system development. It tries to provide techniques, methods, and tools for extracting, specifying, deliberating, and verifying software system requirements. As a result, RE is a critical stage in the development of high-quality systems. It ensures that client requirements are met successfully after the system is constructed. RE begins with gathering requirements and then expressing them. Each requirement must be extracted, allocated, obeyed, fulfilled, justified, and verified precisely.

In the software engineering field, there are various definitions of RE, the earliest of which was provided by Ross and Schoman [5] in 1977: "requirements definition is a careful assessment of the requirements that a system is to fulfill." It has to clarify why a system is desirable according to present and predicted requirements, which could indicate an internal operation or an external effect. It has to respond to which system properties are suitable in this situation. Finally, it has to specify how the system will be created.

Sommerville and Sawyer [7] explicitly characterize RE activities. They define RE as the task of identifying, documenting, and handling a collection of system requirements. These activities are classified as "requirement elicitation, requirement analysis, requirement specification, requirement validation, and requirement management."

Because traditional systems analysis approaches are inefficient when working on safety-critical systems, the reputation of goal-oriented requirements engineering (GORE) methods has grown considerably in recent years [8]. During the requirements stage, traditional approaches regard requirements as simply procedures and data and should not communicate validation to specific top-level contexts in the problematic region. Several approaches focus only on software

system modeling and specification. As a result, they require assistance in understanding the safety-critical systems comprised of the system-to-be and there surroundings. Furthermore, when system functions are discovered to be automated or other responsibilities are revealed to be signified and appraised, traditional modeling and analytical methodologies do not allow for alternative system forms. "Goal-Oriented Requirements Engineering" (GORE) attempts to address these critical challenges [9].

At various levels of abstraction, goals are captured. GORE is the usage of goals for extracting, organizing, identifying, developing, evaluating, documenting, negotiating, and altering requirements. Over the past few decades, this domain has received increasing attention [1]. Goals are regarded as an vital component of the requirements engineering procedure in RE research. For numerous reasons, such as requirements elicitation, specification, analysis, verification, resolution of conflict, and translating informal to formal requirements, this credit has driven the whole flow of research on goal specification, goal-based analysis and goal modeling.

Comprehending, analyzing, and processing huge informal requirements into a tangible system may be required when developing complicated technical systems such as urban subway control systems. These requirements might be of many types, such as security, performance, functionality, compatibility with current systems, and so on. In this article, KAOS is used to create a goal-oriented model of an urban subway control system as a case study to make sure that the real system matches the desired system.

The remainder of the article is structured as follows: Section 2 correlates our method with related works; Section 3 presents the urban subway control system; Section 4 discusses the outcomes of applying KAOS goal-oriented modeling to an urban subway control system; and eventually, in Section 5, we wrap up the paper.

B. Related Work

F. Semmak et al. [10] the KAOS goal-oriented method was used to enhance requirements engineering in the framework of the Cycab domain. Cycab is a completely automatic community transportation vehicle. They further improved the KAOS technique to address variability concerns. This addition enables them to develop a goal model for unpredictability. Furthermore, they verified their goal model with a software prototype based on an unfinished version of the Cycab application framework.

C. Ponsard et al. [9] used the KAOS GORE process to extract requirements and supervise mission-critical systems. The model they offered is an excerpt from the major railroads' signaling specifications. Their initial work documentation was based on a current review of state machine charts with safety standards. They utilized the FAUST toolset for requirement analysis, verification, and validation.

E. Dubois et al. [11] in the framework of reactive systems, they outlined three precise and interrelated modeling tasks during the requirements engineering phase. Furthermore, they demonstrated how the formal languages Timed Automata, Albert II, and KAOS facilitate the aforementioned three tasks. They utilized the i* framework to build a complex model and connect several formal models. They used a tiny process control system to demonstrate their approach.

M. Wilson and K. Wnuk [12] described the drawbacks of the goal model, namely that they do not obtain contextual information, the active agent's knowledge, or another active agent who obtains the goal for fulfillment. As a result, they expanded the goal model with intents and a context frame in their paper to define more contextual information about goals and fulfillment techniques.

C. Ponsard et al. [13] emphasizing a model-based strategy to offer effective assistance during the risk assessment step. They suggested a goal-oriented meta-model of the simple automobile control sub-system for obtaining automotive properties and system features to analyze the effect of catastrophic situations, detect threats, and evaluate their viability.

L. Kadakolmath and Umesh D. R [14] articulated the i* model in the framework of the urban railway IXL system. To solve variability issues and combine early and late requirements, they adopted TGRL, an i*-based GORE language. The created model may also be observed and assessed with jUCMNav. Formal Tropos is used to specify the safety criteria that must be proven. It is regarded as the i* model's formal specification language.

L. Kadakolmath and Umesh D. R [15] a thorough assessment of semi-formal and formal methodologies in the framework of a smart mass transit system was given. They also reviewed the most significant problems that are encountered while implementing computer-based IXL systems. They demonstrated how the use of B formal language and the combination of B with KAOS have increased trust in the use of formal and semi-formal approaches in the urban railway industry.

C. Urban Subway Control System

Rail Rapid Transit (RRT) is a well-established and widely used urban transit system. It carries a large amount of passengers quickly. RRT is a subway system of transportation that operates at high speeds on permanent guideways in metropolitan areas. With technological advancement, the RRT system is now entirely reliant on "Intelligent Transport System" (ITS) technology. Hence, RRT is now referred to as a "Smart Mass Transit System."

The adoption of computerized IXL systems in urban subways resulted in automatic and autonomous trains as well as a rise in demand for highly regular trains with regular stops. However, with technical developments in telecommunications and information technology, the Rail Safety Improvement Act of 2008 proposes a "Positive Train Control" (PTC) system for subway rail [16]. PTC is a combination of cutting-edge technology designed to reduce mishaps triggered by human error, locate malfunctions, malfunctions of equipment, and other types of train operator errors. PTC attempts to prevent train-on-train accidents, detours generated by excessive speed, and early track derailments.

The "Communication-Based Train Control" (CBTC) system is an extremely advanced train control and signaling system that is currently a favored technological solution for smart mass transit systems such as subways all over the globe. CBTC systems are often used to monitor autonomous subway and suburban trains. CBTC requires train information to be delivered to a centralized zone, which in turn transmits the information to all system entities. To track train location and speed, this system employs a "Global Positioning System" (GPS) or transponder tags [16], [17].

For handling traffic and subway control, the CBTC transmits signals between trains and trackside objects. It also detects the train's exact location more precisely. This results in a well-organized and safe system for monitoring railway passenger traffic and reducing train intervals. Not only do these control systems assure subway safety, but they also integrate, interact, and automate areas of operation, gather passenger data, and inspect it. As a result, the precise efficacy and comprehensive safety guarantee of these control systems are crucial. The primary causes of control system malfunctions include faulty requirements specifications, design mistakes, wrong implementations, and human testing, which must be avoided with a high-level degree of certainty. Furthermore, in today's smart mass transit systems, it is important to fulfill not only safety standards but also data precision and operational correctness. For all of these factors, a KAOS-based goal-oriented model is employed in this article to specify and analyze the requirements of an urban subway control system.

D. KAOS Methodology

GORE is concerned with the behaviors that lead to the development of system requirements. The key behaviors that are often included in GORE methods are goal extraction, goal augmentation, and numerous types of goal evaluation, as well as the assignment of goal fulfillment responsibilities to actors. A goal, as defined by Axel van Lamsweerde [18], is "an objective that the system should achieve through the cooperation of agents in the software-to-be and the environment."

GORE depicts the provided system and its surrounding environment as a group of active agents called "actors or stakeholders." Each actor could limit their behavior in order to verify the constraints. The actors in the subway control system detailed in this paper are track circuits, an automatic train controller, a train driver, an interlocking system, and so on. A goal is an intent that the system should attain with the help of actors in the system-to-be and its surroundings. A *requirement* is a goal. To achieve this goal, a system relies on only one active agent in the software system, and it is also an agent's obligation. For instance, "ejection of train doors when alarm rings" is a goal whose accomplishment is reliant on the actor 'train door actuators'. An *expectation* is a goal as well. A system relies on only one active agent in the system context to achieve this goal. For instance, "the green signal should not be turned on until the trap points have been set to the proper position" is a goal whose achievement is dependent on the actor's 'interlocking system'. An *assumption* implies a goal whose achievement is dependent on the achievement of additional goal. *Softgoals*, as opposed to goals, are non-functional requirements for which achievement is not clearly specified. For instance, safety is a non-functional requirement, and certain tasks have a positive influence on it while others have an adverse effect. KAOS places a greater emphasis on goal fulfilment, with fewer emphasis on softgoals and assumptions.

The **KAOS** is a GORE process that includes a comprehensive set of formal evaluation methodologies. KAOS is an acronym that means "Knowledge Acquisition in Automated Specification" [19] or "Keep All Objects Satisfied" [20]. KAOS is portrayed as a multi-paradigm framework that permits various stages of extraction and analysis to be combined [10]. Such as qualitative for deciding between possibilities, semi-formal for organizing and modeling goals, and formal

when more accurate analysis is necessary. As a result, the KAOS goal-oriented language integrates system agents, assumptions, objects, and functions; semantic frameworks for abstract goal modeling; state-based specifications for system functions; and linear-time temporal logic for goal and object definition. For every construct, the KAOS language is divided into two levels: the outside graphical semantic layer and the inside formal layer. The qualities of the object and goal constructs, as well as their relationships with other concepts, are provided in the outer graphical semantic layer, while formal definitions are provided in the inner formal layer.

A goal is described in KAOS as a "prescriptive statement of intent about some system whose satisfaction, in general, requires the cooperation of some of the agents forming that system" [1]. Goals in KAOS can define either functional or non-functional services. Goals in KAOS are systematized using AND/OR refinement hierarchies, and goal refinement is completed when each subgoal is allocated to a different agent. As a result, goals are overseen and controlled by an agent. A goal under an agent's responsibility in a software system is termed a *requirement*, while a goal under an agent's responsibility in a system's environment is called an *expectation*. The lightweight goal specification is stated in temporal logic, utilizing goal definition patterns such as achieve, avoid, maintain, cease, and optimize. Additional goals in KAOS include information goals, satisfaction goals, and accuracy goals. Satisfaction goals are functional goals that are linked to the fulfillment of an agent's requests. Information goals are functional goals linked to informing an agent about object states. Finally, accuracy goals are non-functional goals that are connected to representing the condition of the monitored or regulated objects in the system context.

E. Result and Discussion

The primary purpose of developing the KAOS goal model of an urban subway control system is to offer efficient commuter transportation from one location to another. The KAOS goal model is developed using the objectiver tool [21]. Figure 1 shows the main goal of subway system and its refinement. In Figure 1, a goal is represented by using a parallelogram symbol, and refinements of a parent goal are represented by using yellow circles. The checkmark in the circle repre-sents 'and decomposition' of subgoals, that is all three subgoals should be fulfilled to accomplish the parent goal. The pattern in Figure 1 is read as *"To satisfy the goal effi-cient transportation of urban rail commuters, the control system must provide com-fortable transportation, safe and secure transportation, and rapid transportation services."* To achieve the parent goal *"Efficient Transportation of Train Passengers,"* it is refined into three subgoals as listed below.

Comfortable Transportation: to fulfill this goal urban rail should accelerate or decelerate easily. Train arrival information must be shared in time with the commuters at a railway station. Train departure, which station will arrive next stop and so on are well communicated with commuters inside a train.

Safe and Secure Transportation: to fulfill this goal the possibility of accidents should drop below the threshold enforced by the safety guidelines. For example, the safety distance among two trains travels along with each other must be adequate to avoid the rear train from colliding with the front train. On a specific

block, the speed of a train may never go beyond the speed limit of that block. If a block entry signal is set to stop then a train never enter a block. If a train is moving then its doors must be locked.

Rapid Transportation: to fulfill this goal the headway between trains must be reduced by avoiding excessive delays and running trains at high frequency.

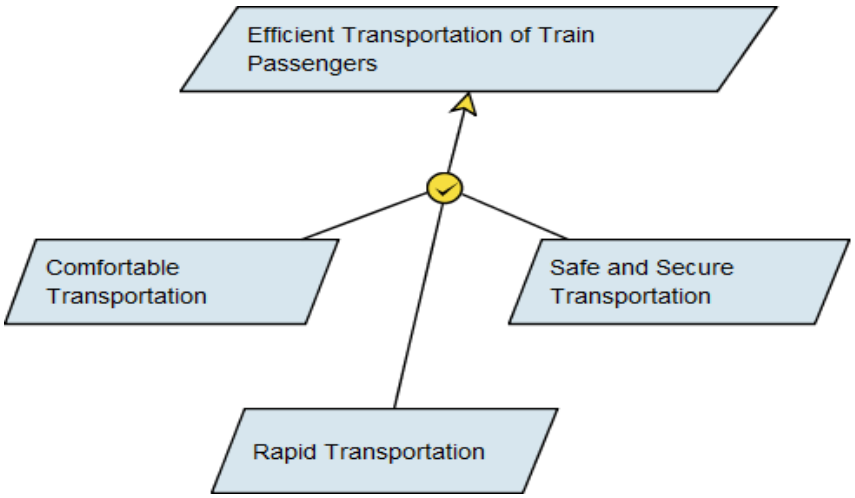

Figure 1. Generic goal 'Efficient Transportation of Train Passengers.'

The comfortable transportation child goal in Figure 1 is further decomposed into subgoals as exposed in below Figure 2. The pattern in Figure 2 is read as *"To satisfy comfortable transportation service, the commuters must reserve a seat easily, and control system must provide boarding and alighting information in time."*

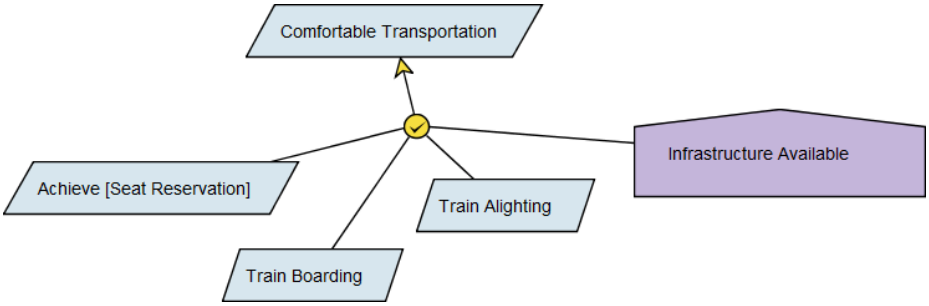

Figure 2. Generic goal 'Comfortable Transportation.'

In Figure 2, *'Infrastructure Available'* is a domain property. To satisfy comfortable transportation, infrastructure to provide comfortable transportation must be available. In Figure 2 it is shown in the pentagon symbol. Properties that are relevant to the application domain are called domain properties. There are two kinds of domain properties. The first one is domain invariants and the second one is domain hypotheses. Domain invariants are descriptive declarations regarding the environment, likely to hold unchangeably in each state of the domain object. For example, *'train doors should open if and only if train speed is zero.'* Domain hypotheses are descriptive declarations fulfilled by the system environs and

subject to change. For example, *'an urban railway control system has at least one display-based interface to provide information to commuters.'*

The safe and secure transportation child goal in Figure 1 is further decomposed into subgoals as shown below Figure 3. The pattern in Figure 3 is read as *"To satisfy safe and secure transportation service, the control system must maintain train doors closed while a train is moving until alarm rings, and it must avoid train collision and de-railment, and train speed should be below its block limit."*

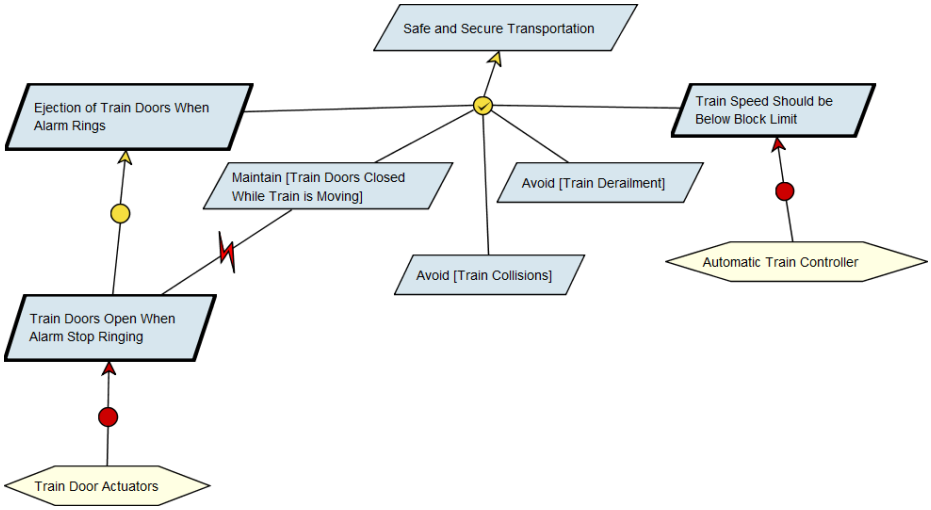

Figure 3. Generic goal 'Safe and Secure Transportation.'

In Figure 3 some requirements have been used. A requirement is a goal in a software system that is accountable to a single agent and is represented by thick-bordered parallelograms. For example, *'ejection of train doors when alarm rings,' 'train speed should be below block limit,'* and *'train doors open when alarm stop ringing.'* Requirements can be further refined as shown in Figure 3. Agents are stakeholders symbolized as pink boxes with angle corners. For example, automatic train controllers and train door actuators. The responsibility relationship is symbolized as red circles, that link an agent to a requirement for which the agent is responsible. Figure 3 also shows a conflict between a goal train door closed while moving and re-quirement train doors open when the alarm stop ringing.

The rapid transportation child goal in Figure 1 is further decomposed into subgoals as shown in below Figure 4. The pattern in Figure 4 is read as *"To satisfy rapid transportation service, the control system must achieve a high frequency of trains, maintain worst case stopping distance between trains and avoid crowding."*

Avoid train collisions child goal in Figure 3 is further decomposed into subgoals as shown in below Figure 5. The pattern in Figure 5 is read as *"To avoid collisions, the control sys-tem must avoid two trains occupying on the same block, and maintain worst case stopping distance."*

In KAOS, a specified goal can be found on several diagrams to refine several higher-level goals, because the goal model is a directed graph. For instance, the goal *'maintain the worst-case stopping distance between successive trains until the train reaches its destination,'* appears in both Figure 4 and Figure 5. It contributes to the goal of rapid transportation and avoids train collisions. In Figure 5

requirement *'train should stop at block entry if a red signal is on,'* is a variant and it should be satisfied by either train driver or automatic train controller. In KAOS, variability can be denoted by using alternate responsibility allocations of goals to agents, and OR goal decompositions.

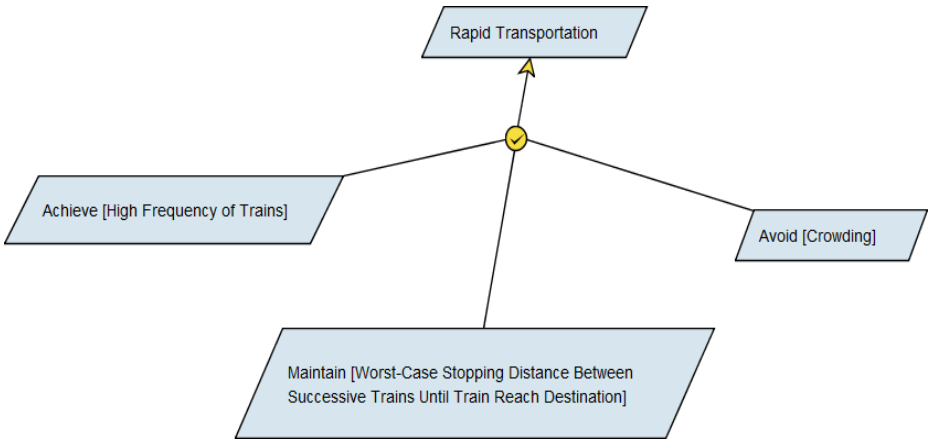

Figure 4. Generic goal 'Rapid Transportation.'

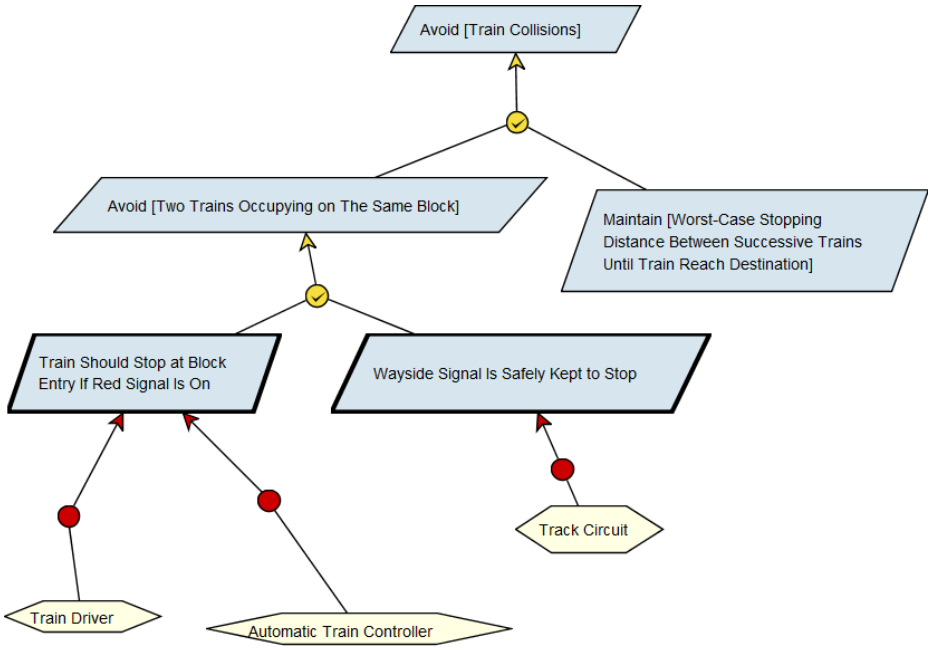

Figure 5. Generic goal 'Avoid Train Collisions.'

Avoid train derailment child goal in Figure 3 is further decomposed into subgoals as shown in below Figure 6. The pattern in Figure 6 is read as *"To avoid train derailment, the control system must avoid sideswipes, avoid unauthorized train for entering into mainline from siding line, and the green signal should not be on until trap points have been set to a proper position, and regular and consistent maintenance of rail infrastructure is required."*

In Figure 6 a goal *'green signal should not be on until the trap points have been set to a proper position,'* is an *expectation*. It is a goal in a system's environment that

is under the accountability of a single actor. They do not need to be refined further, and they are symbolized as thick-bordered yellow-colored parallelograms. Expectation assignment to some agents is symbolized by using pink circles.

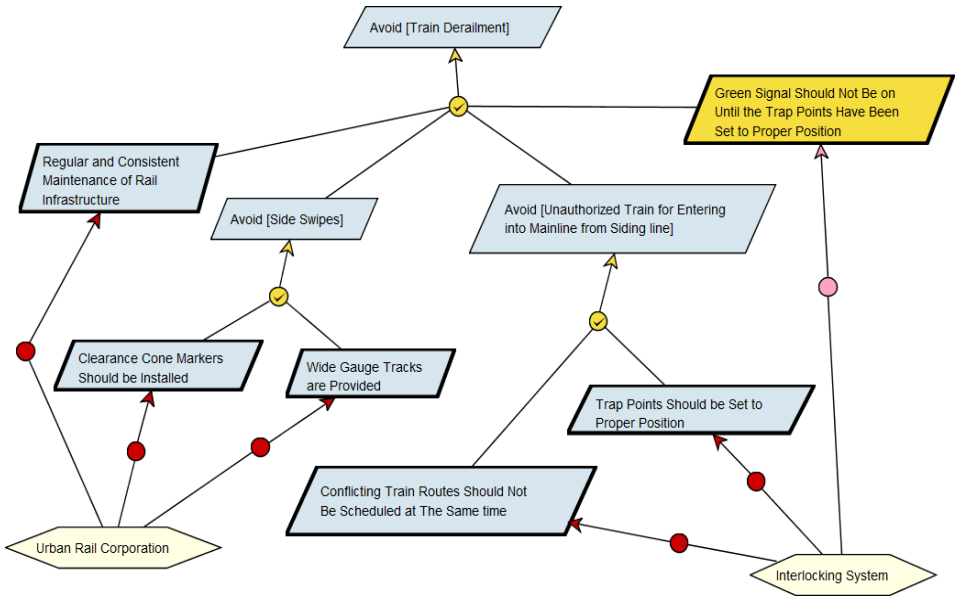

Figure 6. Generic goal 'Avoid Train Derailment.'

Maintain worst-case stopping distance between successive trains until train reach destination in both Figure 4 and Figure 5. is further decomposed into subgoals as shown in below Figure 7. The pattern in Figure 7 is read as *"To maintain worst case stopping dis-tance between successive trains until they reach a destination, the control system must maintain safe acceleration commands and supervise train acceleration."*

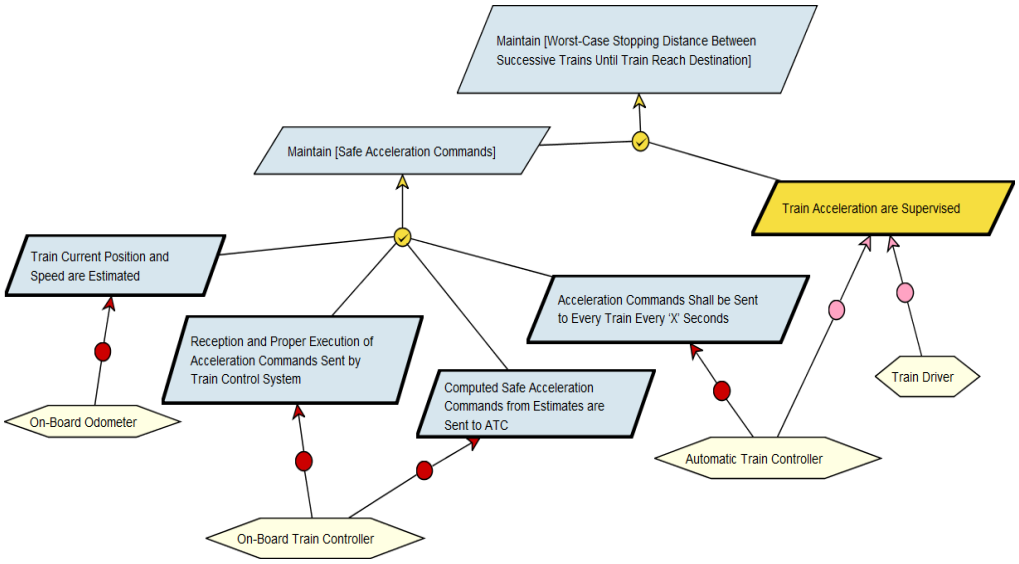

Figure 7. Generic goal 'Maintain Worst-Case Stopping Distance.'

Handling Obstacles

Obstacles are circumstances in which a requirement, an expectation, or even a goal, is violated. The obstacle is supposed to *'obstruct'* a requirement, an expectation, or a goal. Handling obstacles is extremely crucial for safety-critical systems. It permits analysts to recognize and deal with circumstances to give a strong new requirement to prevent or decrease the effects of obstacles.

An assertion that is reliable with the domain model is an obstacle to achieving a goal, but the objective's negation is the logical result of the model composed by an assertion and the domain model. As soon as obstacles are recognized, they can be purified by using AND or OR decomposition. Once the obstacles are handled successfully the resulting software system is more reliable.

For example, in Figure 8, An obstacle *'Red Signal is on and Train not Stopped at Block Entry,'* to the goal *'Train Should Stop at Block Entry if Red Signal is on,'* occurs when the train is not stopped at block entry even if a red signal is on. In Figure 8, obstacles are shown in orange-colored parallelograms. The obstruction link shown in the orange arrow is used to link obstacles to the goals they obstruct. Refinement of obstacles is accomplished in the same way we refine goals, but obstacles are most often *'OR-refined'* while goals are generally *'AND-refined.'* In Figure 8, also the new requirements are linked to the obstacle they resolve by using the resolution link shown in the green arrow. For example, if the signal is not visible then the train driver has to stop the train and wait for the guard signal.

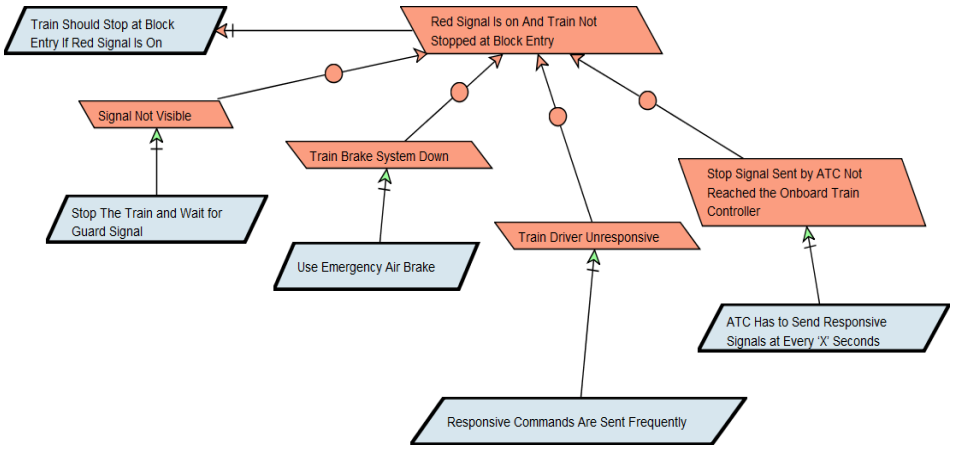

Figure 8. Obstacle 'Red Signal is on and Train not Stopped at Block Entry.'

F. Conclusion

In this article, the authors discussed GORE in the framework of an urban subway control system. Safety-critical systems, such as urban subway control systems, necessitate identifying, extracting, organizing, assessing, documenting, and revising requirements for the concrete system. In this article, they used the KAOS approach to offer appropriate, complete, unambiguous, and consistent requirements in terms of goals. The authors also addressed issues such as conflict management, variability, and obstacle detection. An objectiver tool is used to construct a KAOS-based goal-oriented model. In future enhancements, the authors intend to formalize the goal model using first-order linear-time temporal logic formulae and model-check these requirements.

G. References

[1] A. van Lamsweerde, "Goal-oriented requirements engineering: a guided tour," in *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, Toronto, Ont., Canada: IEEE Comput. Soc, 2000, pp. 249–262. doi: 10.1109/ISRE.2001.948567.

[2] A. van Lamsweerde, *Requirements engineering: from system goals to UML models to software specifications*. Chichester, England ; Hoboken, NJ: John Wiley, 2009.

[3] T. Hall, S. Beecham, and A. Rainer, "Requirements problems in twelve software companies: an empirical analysis," *IEE Proc. - Softw.*, vol. 149, no. 5, p. 153, 2002, doi: 10.1049/ip-sen:20020694.

[4] A. Finkelstein and J. Dowell, "A comedy of errors: the London Ambulance Service case study," in *Proceedings of the 8th International Workshop on Software Specification and Design*, Schloss Velen, Germany: IEEE Comput. Soc. Press, 1996, pp. 2–4. doi: 10.1109/IWSSD.1996.501141.

[5] D. T. Ross and K. E. Schoman, "Structured Analysis for Requirements Definition," *IEEE Trans. Softw. Eng.*, vol. SE-3, no. 1, pp. 6–15, Jan. 1977, doi: 10.1109/TSE.1977.229899.

[6] T. E. Bell and T. A. Thayer, "Software Requirements: Are They Really a Problem?," in *Proceedings of the 2nd International Conference on Software Engineering*, in ICSE '76. Washington, DC, USA: IEEE Computer Society Press, 1976, pp. 61–68.

[7] I. Sommerville and P. Sawyer, *Requirements engineering: a good practice guide*. Chichester, Eng. ; New York: Wiley, 1999.

[8] A. van Lamsweerde, "Goal-oriented requirements engineering: a guided tour," in *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, Toronto, Ont., Canada: IEEE Comput. Soc, 2000, pp. 249–262. doi: 10.1109/ISRE.2001.948567.

[9] C. Ponsard, P. Massonet, A. Rifaut, J. F. Molderez, A. van Lamsweerde, and H. Tran Van, "Early Verification and Validation of Mission Critical Systems," *Electron. Notes Theor. Comput. Sci.*, vol. 133, pp. 237–254, May 2005, doi: 10.1016/j.entcs.2004.08.067.

[10] F. Semmak, R. Laleau, and C. Gnaho, "Supporting variability in goal-based requirements," in *2009 Third International Conference on Research Challenges in Information Science*, Fez, Morocco: IEEE, Apr. 2009, pp. 237–246. doi: 10.1109/RCIS.2009.5089287.

[11] E. Dubois, E. Yu, and M. Petit, "From early to late formal requirements: a process-control case study," in *Proceedings Ninth International Workshop on Software Specification and Design*, Ise-Shima, Japan: IEEE Comput. Soc, 1998, pp. 34–42. doi: 10.1109/IWSSD.1998.667917.

[12] M. Wilson and K. Wnuk, "Towards Multi-context Goal Modeling and Analysis with the Help of Intents," in *2018 IEEE 8th International Model-Driven Requirements Engineering Workshop (MoDRE)*, Banff, AB: IEEE, Aug. 2018, pp. 68–72. doi: 10.1109/MoDRE.2018.00015.

[13] C. Ponsard, V. Ramon, and J.-C. Deprez, "Goal and Threat Modelling for Driving Automotive Cybersecurity Risk Analysis Conforming to ISO/SAE 21434:," in *Proceedings of the 18th International Conference on Security and Cryptography*,

Science and Technology Publications, 2021, pp. 833–838. doi: 10.5220/0010603008330838.

[14] L. Kadakolmath and D. R. Umesh, "i*-Based Goal-Oriented Modeling and Requirements Specification of an Urban Railway Interlocking System," *J. Sci. Res.*, vol. 66, no. 02, pp. 30–39, 2022, doi: 10.37398/JSR.2022.660205.

[15] L. Kadakolmath and D. R. Umesh, "A Survey on Formal Specification and Verification of Smart Mass Transit Railway Interlocking System," *Int. J. Saf. Secur. Eng.*, vol. 11, no. 6, pp. 671–682, Dec. 2021, doi: 10.18280/ijsse.110607.

[16] J. C. Peters and J. Frittelli, "Positive Train Control (PTC): Overview and Policy Issues," PPRI Digital Library, R42637, 2012. [Online]. Available: http://docs.lib.purdue.edu/gpridocs/10

[17] L. Rakesh and L. Kadakolmath, "Modeling and formal verification of SMT rail interlocking system using PyNuSMV," in *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad: IEEE, Mar. 2018, pp. 1–8. doi: 10.1109/RAIT.2018.8388983.

[18] A. Van Lamsweerde, "Requirements engineering in the year 00: a research perspective," in *Proceedings of the 22nd international conference on Software engineering - ICSE '00*, Limerick, Ireland: ACM Press, 2000, pp. 5–19. doi: 10.1145/337180.337184.

[19] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Sci. Comput. Program.*, vol. 20, no. 1–2, pp. 3–50, Apr. 1993, doi: 10.1016/0167-6423(93)90021-G.

[20] A. van Lamsweerde and E. Letier, "From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering," in *Radical Innovations of Software and Systems Engineering in the Future*, M. Wirsing, A. Knapp, and S. Balsamo, Eds., in Lecture Notes in Computer Science, vol. 2941. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 325–340. doi: 10.1007/978-3-540-24626-8_23.

[21] Respect-IT, "Objectiver." in The power tool to engineer your Technical and Business Requirements. Respect-IT, 2012. [Online]. Available: http://www.objectiver.com/