Requirements Validation of Event-Driven Supply Chains Using Model-Based Systems Engineering

Amirali Amiri
Automation Systems Group
Institute of Computer Engineering
TU Wien, Vienna, Austria
amirali.amiri@tuwien.ac.at

***Abstract*—Supply Chain Management (SCM) plays a vital role in business operations, necessitating careful designing and planning. Emerging technologies, such as the Internet of Things (IoT), facilitate real-time, event-driven monitoring of supply chain performance. However, integrating IoT into SCM presents significant challenges for system architects, particularly in validating requirements due to the diversity of IoT technologies and communication frameworks. Model-Based Systems Engineering (MBSE) offers a structured approach to system design by providing models and views, along with tools that automate essential processes. In this paper, we propose an MBSE-based approach for requirements validation in event-driven supply chain management. Additionally, we introduce tool support to enhance usability. An architectural overview demonstrates how system designers can customize the tool to meet their specific requirements. Our approach is designed for system architects, supporting iterative modeling until the requirements are validated as illustrated through a sample case study.**

***Index Terms*—Supply-Chain Management, Internet of Things, Event-Driven, Model-Based Systems Engineering**

I. Introduction

Supply Chain Management (SCM) is essential for efficient business operations [1], [3]. Advancements in technology, particularly the Internet of Things (IoT), enable real-time, event-driven monitoring of supply chain performance, such as detecting and reporting facility malfunctions [11], [14]. However, the adoption of IoT in SCM remains challenging, as system designers must navigate a diverse range of technologies and acquire expertise in their implementation and integration.

A very important challenge is requirements validation, when using the diverse landscape of IoT technologies, frameworks, patterns and standards. For example, there are many different patterns of event-driven communication, e.g., using messaging with MQTT brokers [13], gateways [5], or the publish subscribe pattern [17]. Each of these design choices comes with specific requirements, which can be sometime contradicting. It is upon the architects to ensure that the system model fulfills the requirements.

Established methodologies, such as Model-Based Systems Engineering (MBSE) [6], are instrumental in optimizing the design process by introducing structured abstractions through models and views. These abstractions help system designers manage complexity, improve traceability, and maintain consistency throughout the development lifecycle. Additionally, supporting tools enhance the efficiency and accessibility of MBSE by automating various tasks, reducing manual workload, and minimizing errors.

In order to illustrate, artifact generators can automatically create documentation and provide insights into the results of automated requirements validation. This automation not only accelerates development but also supports iterative refinement and validation, ensuring that system requirements are met. MBSE, when combined with appropriate tool support, offers a comprehensive framework for tackling the challenges associated with modern system design and integration. Thus, we set out to answer the following research questions:

RQ1: *How can MBSE assist the requirement validation of event-driven supply-chain management systems?*

RQ2: *What is the architecture of tool support that facilitates requirements validation of event-driven supply chains?*

The main contributions of this paper are as follows. We introduce an MBSE-based approach specifically designed for requirements validation in event-driven supply chain management systems. To achieve this, we define metadata that describe these systems, serving as tags within Systems Modeling Language (SysML) 2.0[1] model instances. Our framework automatically transforms these instances into graph representations and stores them in a Neo4j database[2]. We then perform graph-based validation to verify requirements and detect any violations. Additionally, we present a prototype tool that supports the practical application of our approach, available as an online artifact[3]. To further illustrate the adaptability of our framework, we provide architectural insights into our tool, demonstrating how system designers can customize and extend it to meet their specific requirements.

II. State of the Art

Adama et al. [2] argue that the digital transformation is reshaping supply-chain management, improving efficiency through technologies like AI, big data, and blockchain. This paper examines its economic impact, highlighting shifts from traditional models to digital ecosystems to understand stakeholder interactions. Like our study, it enhances visibility,

[1]https://www.omg.org/spec/SysML/2.0/Beta1
[2]https://neo4j.com
[3]https://zenodo.org/uploads/14974098 DOI:10.5281/zenodo.14974098

© 2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works

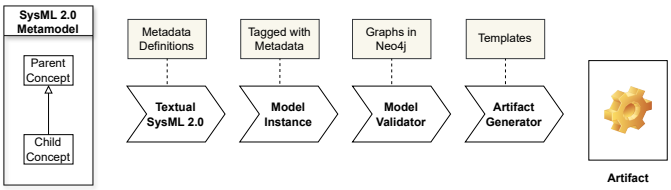

Fig. 1: The Workflow of Model-Based Systems Engineering

tracking, and agility, while automation optimizes resources and reduces disruptions. The authors indicate that challenges like system integration and trends such as IoT technologies must be considered, which is the focus of our paper.

Abdul-Azeez et al. [1] mention that optimizing SCM is crucial for improving efficiency, reducing costs, and enhancing customer satisfaction. They streamline SCM by integrating procurement, manufacturing, logistics, and inventory management into SAP S/4HANA, an enterprise-resource-planning suite. It leverages real-time data, advanced analytics, and automation to improve visibility, decision-making, and agility. The key features of their suite, e.g., predictive analytics and IoT integration, optimize demand forecasting, inventory, and logistics. Likewise, Ikevuje et al. [10] explore how IoT and data analytics improve supply chain efficiency, reduce costs, and enhance performance. They address challenges like poor visibility, inventory issues, delays, and risk management. Key recommendations include investing in IoT infrastructure, ensuring data security, and promoting collaboration. The findings emphasize IoT and data analytics in creating resilient and agile supply chains. Unlike our study, these papers do not provide a model-based systems engineering approach to abstract the system design and generate code automatically.

Goli et al. [9] argue that supply-chain network design is crucial in today's competitive landscape, with transportation costs rising relative to manufacturing companies' income. Both strategic and tactical decisions are key in shaping effective supply chains. This research presents a flexible and sustainable IoT-based supply chain with integrated logistics, involving suppliers, producers, distribution centers, customers, and repair/disassembly centers. Flexibility is achieved by allowing direct shipments to customers from distribution centers or manufacturing plants, with an IoT system managing both direct and indirect deliveries. A mixed-integer linear programming model is used to represent the problem. However, a learning curve is required to use their approach. In our study, we provide a simplification via MBSE and prototypical tool support to make our approach easy to use. In the following, we list the related studies focusing on MBSE.

Vogel-Heuser et al. [19] highlight how the properties and environmental dependencies of industrial systems influence automated production performance. These properties, often in supplier documents like manuals or catalogs, are represented using SysML profiles and disciplinary views, supported by a metamodel for clarity. To address the static nature of profiles, they employ business-process-modeling notation to showcase SysML's dynamic workflow benefits. Earlier, Vogel-Heuser et al. [18] introduced SysML for Automation (SysML-AT), a tailored SysML profile for MBSE in manufacturing automation. It integrates functional and non-functional requirements, software, and hardware, enabling automated software generation for run-time environments. A prototype tool embeds adapted SysML parametric diagrams into industrial software development, allowing in-model debugging. Case studies and usability tests validate its effectiveness for MBSE in manufacturing.

De Saqui-Sannes et al. [7] explore the evolution of Systems Engineering from document-centric to model-centric MBSE approaches. While earlier works outline MBSE's benefits and limitations, this paper focuses on providing industry professionals with criteria for selecting MBSE languages, tools, and methods, expanding beyond common techniques like SysML. Kharatyan et al. [12] discuss the impact of digitization on technical systems, such as autonomous driving, emphasizing the challenges posed by increased complexity and interconnectivity. They advocate for MBSE approaches to address these challenges, stressing the need to incorporate security considerations early in system design to ensure reliability.

Cederbladh et al. [6] indicate that there's been a shift from document-based to model-based development in systems engineering due to increased complexity and the need for digital workflows. They propose that MBSE is essential, providing early models for analysis and automated tasks. However, there's no common approach for early Verification and Validation (V&V) of system behavior in MBSE. They performed a systematic literature review with 149 of 701 relevant publications on early V&V in MBSE. Their findings show early V&V aims to ensure design quality before implementation, with SysML as the standard language. The authors conclude that V&V solutions vary, often targeting functional properties and being context-specific, with common issues in readiness, simplifications, and tool integration.

Kharatyan et al. [12] highlight the growing integration of information and communication technologies in technical systems due to digitization, which offers benefits like autonomous driving but also presents development challenges. To tackle these challenges, MBSE approaches are used to manage the increasing complexity and interconnectivity of products. However, ensuring the reliability of future systems necessitates the early consideration of security aspects. Like our study, the above-mentioned works provide a system model and an approach to support the system design. However, we focus on the interplay of MBSE and IoT, as well as their benefits in SCM. This integrated view is lacking in the literature, also providing support for requirement validation.

III. Approach Details

Approach Overview This section introduces our MBSE approach. The MBSE workflow [15] is outlined in Fig. 1. Utilizing SysML 2.0's textual representation[1], we define metadata for the IoT-based supply-chain management. These metadata are used as tags when creating a *Model Instance*. These instances are transformed into graphs and stored in a Neo4j

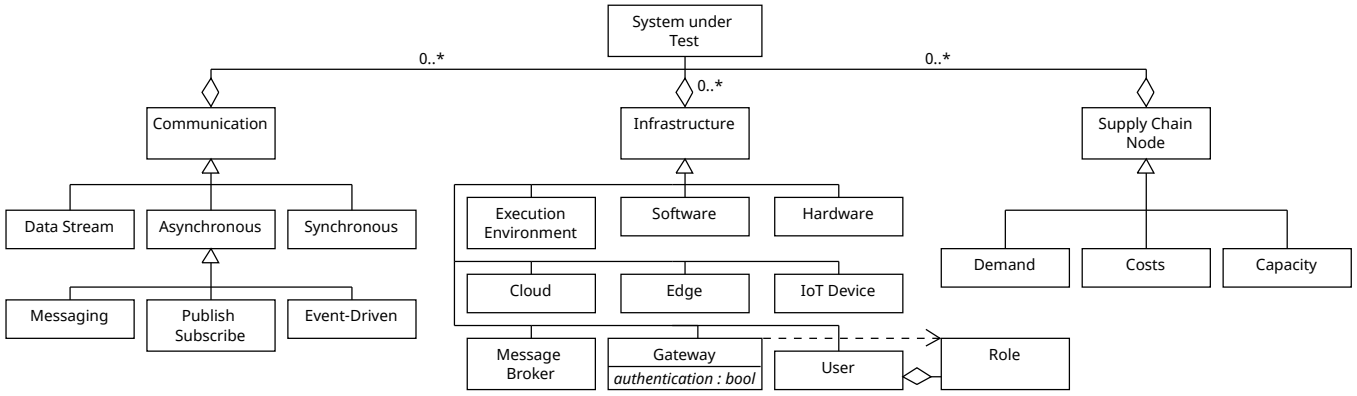

Fig. 2: Metadata of Event-Driven Supply Chain Management

database[2]. A *Model Validator* verifies that system requirements are met. Once validated, the models are sent to an *Artifact Generator* to produce artifacts such as code, test cases, or documentation. The only additional task for architects or system designers is to tag system components with our predefined metadata definitions in SysML 2.0. Our methodology automatically converts the system model to graph representations, validates requirements using graph-based methods, creates a graph visualization, and highlights any system components that violate requirements.

```
package Meta_SupplyChain {

    metadata def SupplyChainNode;
    metadata def Demand :> SupplyChainNode;
    metadata def Capacity :> SupplyChainNode;
    metadata def Costs :> SupplyChainNode;

    metadata def Infrastructure;
    metadata def Edge :> Infrastructure;
    metadata def Cloud :> Infrastructure;
    metadata def IoTDevice :> Infrastructure;
    metadata def Gateway :> Infrastructure;
    metadata def MessageBroker :> Infrastructure;

    enum def UserRole;
    metadata def User :> Infrasructure {
        import Meta_SupplyChain::UserRole::*;
        attribute Role : UserRole;}

    metadata def Communication;
    metadata def Asynch :> Communication;
    metadata def Synch  :> Communication;
    ...
}
```

Listing 1: Excerpt of Metadata Definitions in SysML 2.0

Metadata Our metadata definitions for the supply chain optimization using IoT systems are presented in Fig. 2. A *System under Test* includes *Supply Chain Nodes*, *Infrastructure* and *Communication*. A supply chain node represents a location with supply and demand. These nodes have *Capacity* of production and *Demand* with associated *Costs*. *Infrastructure* represent the underlying IoT system and users. An *Execution Environment* models either virtual or physical environments, e.g., virtual machines, containers, or bare-metal servers. These environments have *Software* and *Hardware* components. The IoT system consists of *Edge*, *Cloud*, *Message Broker* [13], *Device Gateway*, *IoT Device* nodes, and *User* with *Role* for authentication. These node communicate using different patterns of *Communication*. These best-practices are *Data Streaming*, *Synchronous*, and *Asynchronous*, which can be *Event-Driven*, *Messaging* [5], or *Publish Subscribe* [17]. Listing 1 provides a snippet of our metadata.

Supply-Chain Model We model supply chain nodes with specific characteristics, i.e., locations with capacity of production, demands for finished products, and associated costs. The associated costs for each location are, e.g., manufacturing costs for different products, and freight costs for shipping and transport. Listing 2 shows a snippet of the event-driven SCM model. We have defined a composite part *location* that contains subparts. We model multiple IoT devices, e.g., manufacturing and transportation. These devices send monitoring data to a gateway. The gateway sends the data via a message broker to the edge and cloud services for analytics.

```
package SupplyChainModel {
    import Meta_SupplyChain::*;

    part def demand {@Demand;}
    part def supply {@Capacity;}
    part def cost {@Cost;}

    part location {@SupplyChainNode;
      part demand;
      part supply;
      part cost;
    }

    // IoT Infrastructure
    part MQTT {@MessageBroker;}
    part CloudService {@Cloud;}
    part EdgeService {@Edge;}
    part IoTGateway {@Gateway;}
    part Device {@IoTDevice;}
    part UserRole {@User;}
}
```

Listing 2: Supply-Chain Model in SysML 2.0

Graph Representation of Model Instances System designers simply need to import our metadata definitions into a SysML 2.0 textual representation and tag their system components accordingly. Once the system components are tagged, we convert the system model into a graph. This graph

representation reflects the components-and-connectors view of the system [4]. Section V offers an illustrative sample case.

Graph-Based Validation of Requirements We traverse the graph representation of model instances for requirements validation. To illustrate our approach, we provide algorithms to validate three requirements as a proof-of-concept. The first requirement we check is event-driven communication [16] for the IoT infrastructure. For this requirement, we check all paths of the converted system graph to make sure that all communications go through message brokers, i.e., communication is asynchronous. Our approach indicates a violation of system requirements, if there exists a path that the requests can be passed between system parts synchronously without going through a message broker. Algorithm 1 presents our validation of this requirement.

Algorithm 1: Graph-Based Requirement Validation: *IoT Communication must be event-driven.*

Input: $G \leftarrow Graph(IoTModel)$
$violation \leftarrow$ false
foreach *path in G* **do**
  if *path does not include a MessageBroker* **then**
    $violation \leftarrow$ true
    break
  end
end
return $violation$

The second requirement that we validate is exclusion of specific supply-chain nodes. In order to clarify, there exists many scenarios, where there is fault in a supply-chain node, or the manufacturing costs of a specific node has risen due to shortage of supplies. Moreover, IoT devices, e.g., transportation monitoring devices, can indicate a surge in transportation costs for a specific supply-chain node. It is very common for these scenarios to exclude a specific supply-chain node in a specific path. Designers analyse alternative routes to meet demands. Algorithm 2 gives our requirements validation regarding this scenario. In Section IV, we provide further details on prototypical implementation of our algorithms.

Algorithm 2: Graph-Based Requirement Validation: *Exclusion of a specific supply-chain node*

Input: $G \leftarrow Graph(SCMModel)$
Input: $ExcludedNode, StartNode, EndNode$
$violation \leftarrow$ false
foreach *path in G* **do**
  if *path.startNode == StartNode AND path.endNode == EndNode* **then**
    if *path includes ExcludedNode* **then**
      $violation \leftarrow$ true
      break
    end
  end
end
return $violation$

The third requirement validated by our approach is authentication. Namely, gateways must authenticate all requests coming from users. Algorithm 3 presents our algorithm for role-based authentication [16].

Algorithm 3: Graph-Based Requirement Validation: *Requests must be authenticated.*

Input: $G \leftarrow Graph(IoTModel)$
Input: $User$
$violation \leftarrow$ false
foreach *path in G* **do**
  if *path.startNode == User* **then**
    if *path does not include Gateway with authentication* **then**
      $violation \leftarrow$ true
      break
    end
  end
end
return $violation$

Artifact Generation After validating the models, our approach generates a visualization of the system graph along with information about any requirement violations. This step allows system architects to verify that the converted graph accurately represents the system model, ensuring the information provided is correct. Our approach identifies the specific system parts that violate requirements. If the graph is found to be inaccurate, architects or system designers can tag their system components again and rerun the process. Additionally, the metadata definitions can be extended to better meet the needs of different systems. Section V shows our artifact generation.

IV. Tool Support

We provide a prototype tool to demonstrate our approach, accessible as an online artifact in our study[3]. To minimize the learning curve, we use a simplified syntax based on Systems Modeling Language (SysML) 2.0[1]. The tool supports SysML 2.0 concepts such as Parts, Connections, and Metadata. In this context, Parts represent any system entities, such as hosts, hardware, software, configurations, and artifacts. Connections define links between these entities, while tags are used to label parts and connections, facilitating artifact generation.

The architecture of our tool is illustrated in Fig. 3. The frontend is built using React[4], while the backend is implemented as a RESTful API[5] in Python. We used the Model-View-Controller (MVC) pattern [8] with Server-Sent-Events (SSE) [21] to keep frontend views updated. The publish-subscribe model [17] is implemented with Redis[6] for SSE. The backend API gateway publishes data, and the frontend subscribes to topics for rendering. This setup supports real-time frontend

[4]https://reactjs.org
[5]https://restfulapi.net
[6]https://redis.io

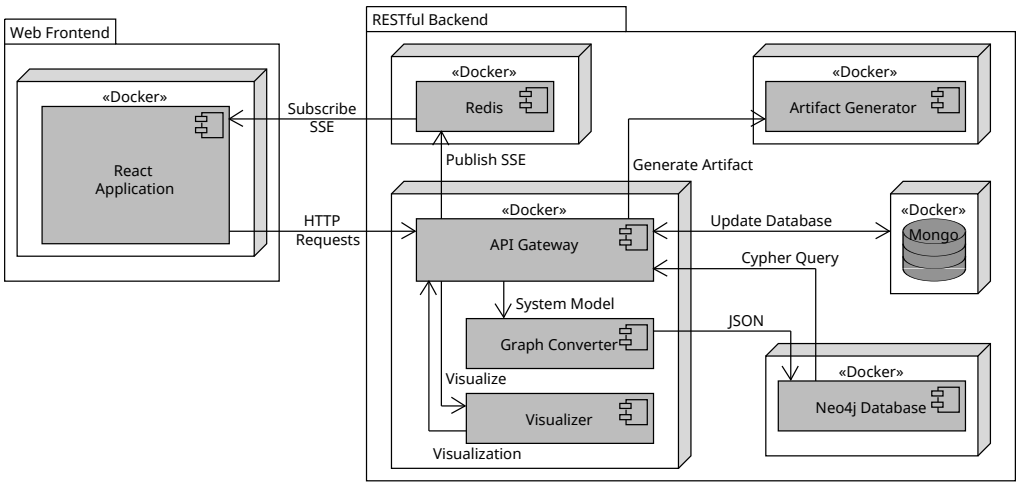

Fig. 3: Tool Architecture Diagram

updates. Visualizations are created with PlantUML[7], and the database is managed with MongoDB[8].

The system model is converted into multiple JSON files and loaded into a Neo4j graph database. We query the graph database using Cypher[2]. Finally, our algorithms in Section III check the results of the queries for requirements validations. The *Artifact Generator* provides a visualization of the converted graph and gives information on the exact parts that violate the requirements. We use the Docker technology[9].

```
package SampleCase {
    import SupplyChainModel::*;

    // Supply-Chain Nodes
    part startNode :> location;
    part middleNode :> location;
    part endNode :> location;

    // IoT Infrastructure
    part broker  :> MQTT;
    part cloud :> CloudService;
    part edge :> EdgeService;
    part gateway :> IoTGateway;
    part admin :> UserRole {Role=Admin;}}

    connect startNode to excludedNode;
    connect excludedNode to endNode;

    connection {@Asynch;} connect Admin to gateway;
    connection {@Asynch;} connect gateway to MQTT;
    connection {@Asynch;} connect MQTT to edge;
    connection {@Asynch;} connect edge to cloud;
    ...
}
```

Listing 3: SysML 2.0 Model of the Sample Case

V. Discussion

We model an illustrative sample case and study different scenarios of the supply-chain management. Listing 2 showed the supply-chain model. The following scenario highlights the importance of the IoT infrastructure and the model-based systems engineering. The MBSE end-to-end approach allows the IoT system to inform of any infrastructure changes as early as possible. Having this information can help decision makers to adapt to the changes rapidly and optimize the supply chain to meet performance requirements.

Our approach is tailored for system designers during an early stages of design, i.e., system architecting. We support iterative modeling until the requirements are validated. This approach is illustrated in the following sections.

Illustrative sample case We model an event-driven SCM system with a central broker in SysML 2.0, where IoT devices, edge and cloud services are modeled. The data of the devices are passed to the edge and cloud services. Listing 3 shows the tagged model instance, and Fig. 4 presents the converted graph of the sample case. We have a simple model of three supply-chain nodes. For better readability, we named them according to their position in the supply-chain model, i.e., *startNode*,

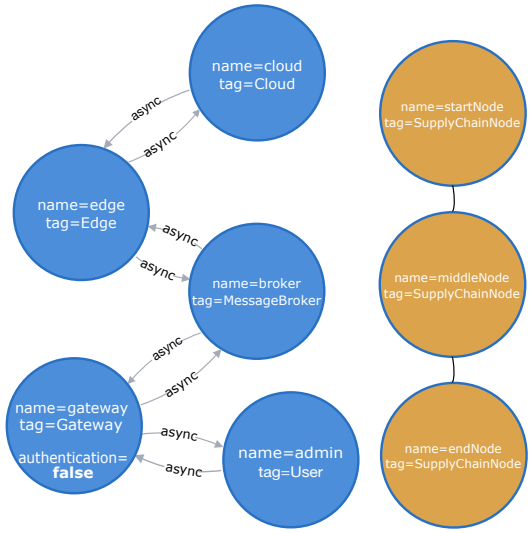

Fig. 4: Converted Graph of the Sample Case

[7]https://plantuml.com
[8]https://www.mongodb.com
[9]https://www.docker.com

Failed the requirements of the event-driven communiaction:
edge -> cloud , cloud -> edge
Failed the requirements of the exclusion node:
startNode -> middleNode -> endNode
Failed the requirements of the authentication:
admin -> gateway

Fig. 5: Results regarding Failed Requirements

middleNode, and *endNode*. Next, we study a scenario to exclude the *middleNode* in the supply chain.

Results Fig. 5 shows the results of our framework regarding the sample case. Our approach informs about the paths that violate the event-driven communication. We can see that even though the connection between *edge* and *cloud* services is tagged as *asynchronous*, but it does not go through a message broker. We can see that our approach informs about a violation on this supply-chain path marked yellow in the system graph. Regarding the authentication requirement, we did not set the authentication in the *gateway* to *true*.

Following the information of our approach, we update the system model so that the *edge* service communicates with the *cloud* service via the *MQTT*. Moreover, we model authentication in the *gateway*. Fig. 6 shows our updated design, and Fig. 7 shows the results of our framework regarding the updated graph. We see that we passed the requirements of event-driven communication and the authentication.

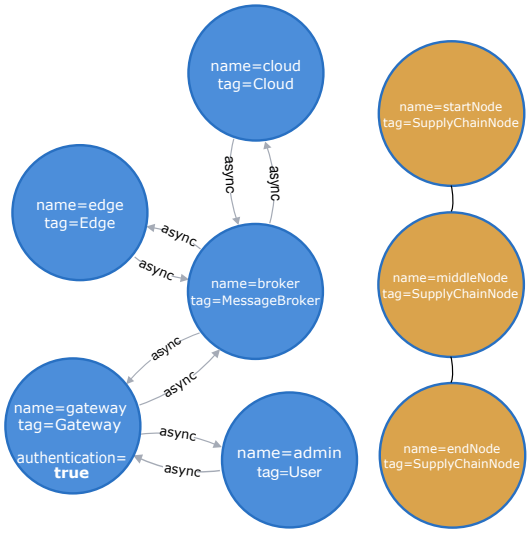

Fig. 6: Updated Graph of the Sample Case

Passed the requirements of the event-driven communiaction.
Failed the requirements of the exclusion node:
startNode -> middleNode -> endNode
Passed the requirements of the authentication.

Fig. 7: Results regarding the Updated Graph

Finally, we connect the *startNode* and the *endNode* of the supply chain directly as shown by Fig. 8. The results of our framework is shown in Fig. 9 passing all requirements. The recommendations of our approach helps the architects to update the system design early in the process.

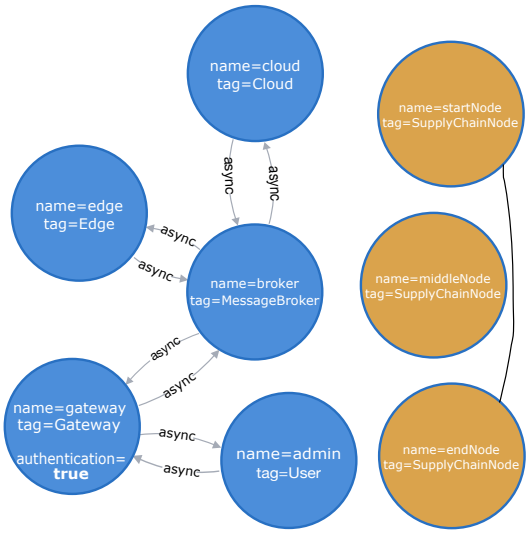

Fig. 8: Updated Supply Chain

Passed the requirements of the event-driven communiaction.
Passed the requirements of the exclusion node.
Passed the requirements of the authentication.

Fig. 9: Results regarding the Passed Requirements

VI. Conclusion and Future Work

In this paper, we proposed an approach to automatically validate requirements of the event-driven supply-chain management. Moreover, we generate artifacts that inform about the violation report. Our study was motivated based on a stakeholder-analysis of industrial automation firms that highlighted the necessity for a cost-effective, all-encompassing approach in this domain. Supply chain management is crucial in business [1], [3]. The new technologies, such as IoT help the efficiency of SCM.

We set out to answer how can MBSE assist the requirement validation of event-driven supply-chain management systems (**RQ1**), and what the architecture of tool support that facilitates requirements validation of event-driven supply chains is (**RQ2**). For **RQ1**, we proposed a metamodel representing the event-driven SCM. Based on this metamodel, we proposed a model-based systems engineering approach using the textual representation of SysML 2.0. For **RQ2**, we implemented a prototypical tool that is available in our online artifact[3]. This tool implements our concepts and converts the system models into graphs. We perform a graph-based validation of requirements and inform the system architects in case of violation. We provided support for three requirements as a proof-of-concept. This paper defines the backbone of our approach and we aim to provide a comprehensive framework in the future work.

REFERENCES

[1] O. Abdul-Azeez, A. O. Ihechere, and C. Idemudia. Optimizing supply chain management: strategic business models and solutions using sap s/4hana. 2024.

[2] H. E. Adama, O. A. Popoola, C. D. Okeke, and A. E. Akinoso. Economic theory and practical impacts of digital transformation in supply chain optimization. *International Journal of Advanced Economics*, 6(4), 2024.

[3] A. A. Akinsulire, C. Idemudia, A. C. Okwandu, O. Iwuanyanwu, et al. Supply chain management and operational efficiency in affordable housing: An integrated review. *Magna Scientia Advanced Research and Reviews*, 11(2):105–118, 2024.

[4] V. Bertram, S. Maoz, J. O. Ringert, B. Rumpe, and M. von Wenckstern. Component and connector views in practice: An experience report. In *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 167–177, 2017.

[5] H. Cabane and K. Farias. On the impact of event-driven architecture on performance: An exploratory study. *Future Generation Computer Systems*, 153:52–69, 2024.

[6] J. Cederbladh, A. Cicchetti, and J. Suryadevara. Early validation and verification of system behaviour in model-based systems engineering: A systematic literature review. *ACM Trans. Softw. Eng. Methodol.*, 2024.

[7] P. De Saqui-Sannes, R. A. Vingerhoeds, C. Garion, and X. Thirioux. A Taxonomy of MBSE Approaches by Languages, Tools and Methods. *IEEE Access*, 10:120936–120950, 2022.

[8] R. F. García. *MVC: Model–View–Controller*. Apress, Berkeley, 2023.

[9] A. Goli, E. Babaee Tirkolaee, A.-M. Golmohammadi, Z. Atan, G.-W. Weber, and S. S. Ali. A robust optimization model to design an iot-based sustainable supply chain network with flexibility. *Central European Journal of Operations Research*, pages 1–22, 2023.

[10] A. H. Ikevuje, D. C. Anaba, U. T. Iheanyichukwu, et al. Optimizing supply chain operations using iot devices and data analytics for improved efficiency. *Magna Scientia Advanced Research and Reviews*, 2024.

[11] Y. Khan, M. B. M. Su'ud, M. M. Alam, S. F. Ahmad, A. Y. B. Ahmad, and N. Khan. Application of internet of things (iot) in sustainable supply chain management. *Sustainability*, 15(1):694, 2022.

[12] A. Kharatyan, J. Tekaat, S. Japs, H. Anacker, and R. Dumitrescu. Metamodel for safety and security integrated system architecture modeling. *Proceedings of the Design Society*, 1:2027–2036, 2021.

[13] S. Mirampalli, R. Wankar, and S. N. Srirama. Evaluating nifi and mqtt based serverless data pipelines in fog computing environments. *Future Generation Computer Systems*, 150:341–353, 2024.

[14] P. Nanjundan, B. V. James, J. P. George, and A. Tiwari. Iot-enabled supply chain management for increased efficiency. In *2024 International Conference on Trends in Quantum Computing and Emerging Business Technologies*, pages 1–5. IEEE, 2024.

[15] A. L. Ramos, J. V. Ferreira, and J. Barceló. Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(1):101–111, 2012.

[16] C. Richardson. Microservice architecture patterns and best practices. *http://microservices.io/index.html*, 2019.

[17] N. Sai Lohitha and M. Pounambal. Integrated publish/subscribe and push-pull method for cloud based iot framework for real time data processing. *Measurement: Sensors*, 27:100699, 2023.

[18] B. Vogel-Heuser, D. Schütz, T. Frank, and C. Legat. Model-driven engineering of Manufacturing Automation Software Projects – A SysML-based approach. *Mechatronics*, 24(7):883–897, 2014.

[19] B. Vogel-Heuser, M. Zhang, B. Lahrsen, S. Landler, M. Otto, K. Stahl, and M. Zimmermann. Sysml' – incorporating component properties in early design phases of automated production systems. *at - Automatisierungstechnik*, 72(1):59–72, 2024.

[20] C. Wohlin, P. Runeson, M. Hoest, M. C. Ohlsson, B. Regnell, and A. Wesslen. *Experimentation in Software Engineering*. Springer, 2012.

[21] D. Yunga-Yunga, D. Quisi-Peralta, M. López-Nores, R. García-Velez, J. Salgado-Guerrero, and L. Serpa-Andrade. Successful case in the development and implementation of a multi-company inventory management system based on communication protocols remote procedure calls (grpc) and server-sent events (sse). In *Information Technology and Systems*, pages 141–150, Cham, 2024. Springer Nature Switzerland.