VR-SDLC: A Context-Enhanced Life Cycle Visualization of Software-or-Systems Development in Virtual Reality

Roy Oberhauser[0000-0002-7606-8226]

Computer Science Dept., Aalen University, Aalen, Germany
roy.oberhauser@hs-aalen.de

Abstract. As systems grow in complexity, so does their associated lifecycle and with it the need to manage the various elements, relations, and activities involved in the Software (or Systems) Development Life Cycle (SDLC). Various notations for system, software, or process modeling have been specified such as the Systems Modeling Language (SysML), Unified Modeling Language (UML), Business Process Model and Notation (BPMN) respectively, yet due to their two-dimensional (2D) diagram focus, they are ill suited for visualizing a comprehensive contextualized view of the entire systems engineering or software engineering lifecycle. To address this need, the Lifecycle Modeling Language (LML) utilizes a relatively simple ontology and three primary diagrams while supporting extensibility. Yet lifecycle comprehension, analysis, collaboration, and contextual insights remain constrained by current 2D limitations. This paper contributes our Virtual Reality (VR) solution concept VR-SDLC for holistic visualization of SDLC elements, relations, and diagrams. Our prototype implementation utilizing LML demonstrates its feasibility, while a case study exhibits its potential.

Keywords: Virtual Reality, Systems Lifecycle, Software Development Life Cycle, Systems Engineering, Software Engineering, Systems Modeling, Lifecycle Modeling Language (LML), Visualization.

1 Introduction

Software Engineering (SE) and Systems Engineering (SysE) involves lifecycles, yet such lifecycles are rarely if ever formally modeled beyond some higher-level abstraction, nor are technical instantiations, e.g., as processes, concretely used (perhaps because they are subject to frequent variation). Typically engineering, and particularly system or software[1] design, involves modeling the target system-of-interest, where supporting methodologies or notations such as Model-Driven methods (MDX) are well-known and have reached some level of maturity. However, the lifecycle itself is not typically modeled, thus much of what is planned and executed during the associ-

[1] [1]For this article, "systems" development/engineering can be replaced with "software" when hardware development is out-of-scope for some endeavor.

ated lifecycle is not readily accessible to stakeholders in a contextual manner. But with regard to quality, not only the system itself, but the associated processes, methods, techniques, activities, and artifacts utilized during SysE and SE have associated expectations and conformance requirements that play an essential role in achieving some set of qualities (safety, security, reliability, etc.). For example, maturity models (e.g., CMMI), standards (e.g., IEC 61508), guidelines (e.g., MISRA), etc., often focus on preventative process-centric measures to address construction or systemic error risks, since - especially for software-intensive systems - comprehensive verification and validation (V&V) of the final system alone is not viable. Thus, various stakeholders must access, analyze, and collaborate on lifecycle elements. And while the target system and its associated model to be developed can be complex, the lifecycle and surrounding SysE system or environment can often necessarily be even more complex. However, any work performed is often "hidden" in various artifacts and activities by actors, yet not necessarily (easily) tracked and retained. This, in turn, can affect efficiency, effectiveness, traceability, V&V, and compliance. For instance, Automotive SPICE exemplifies such expectations within the automotive industry.

But realizing lifecycle modeling necessitates an appropriate modeling language. The Lifecycle Modeling Language (LML) [1] positions itself as a systems engineer's language for integrating all lifecycle disciplines (program management, systems engineering, testing, deployment and maintenance) into a single framework, such that all stakeholders can model and understand systems. It can be easily translated [2] to other common notations and languages such as BPMN [3], UML [4], SysML [5], etc., and can be used to support holistic traceability through the lifecycle, be it artifacts or activities. Yet when one considers the various disciplines, lifecycle elements, actors, and relations involved in a SysE or SE endeavor (which can be viewed as an open system), its actual complexity (and possibly reactive over-simplification) can make true comprehension across stakeholders challenging. It is interesting to note that the Systems Engineering Vision 2025 from INCOSE foresees systems modeling using immersive technology to share system understanding across stakeholders [6]. Towards supporting stakeholder comprehension and insight, the intent of this paper is to investigate and explore an immersion capability for SDLC utilizing LML.

Our VR-related prior work includes: VR-EA [7] supporting Enterprise Architecture (EA) Archimate models in VR; VR-EAT [8] integrating the tool Atlas for visualizing dynamically-generated EA diagrams; VR-EA+TCK [9] integrating Knowledge Management Systems (KMS) and Enterprise Content Management Systems (ECMS); VR-EvoEA+BP [10] for context-enhanced enterprise models with animated evolution and business process (BP) dynamics; and VR-BPMN [11], VR-SysML [12], and VR-UML [13] visualizing BPMN, SysML, and UML models respectively.

This paper contributes VR-SDLC, our nexus-based VR solution concept for a context-enhanced visualization of lifecycle models for SysE and SE lifecycles. It comprehensively portrays lifecycle model elements, relations, and diagrams enhanced with context and exemplified with LML models. Furthermore, multiple lifecycle projects can be portrayed concurrently to support cross-project or program portfolio analysis and optimizations. Its hypermodeling capability (depiction with other models such as SysML/UML in VR) showcases how both a target model and its lifecycle can be portrayed concurrently in VR. Our realization demonstrates its feasibility, while a case study using sample LML models exhibits its potential.

The paper is structured as follows: Section 2 discusses related work; Section 3 gives a background on LML; our solution concept is described in Section 4; Section 5 details our realization; our evaluation is in Section 6, followed by a conclusion.

2 Related Work

With regard to development lifecycle process standards, ISO offers the ISO/IEC/IEEE 15288 [14] for systems or 12207 [15] for software. However, these exist in natural language and do not provide a machine-processable technical implementation nor modeling language for their process models. Dori [16] laments ambiguous aspects of the 15288 lifecycle standard and how a modeling approach might address these. SysML 1.x lacked an official complete ontology, while SysML version 2.0 is still in beta and thus has limited support.

Work related to lifecycle tool integration includes the Open Services for Lifecycle Collaboration (OSLC) [17], which focuses on specifications for the integration of software development tooling. It can be used for Application Lifecycle Management (ALM), Product Lifecycle Management (PLM), and IT Operations and utilizes the Resource Description Framework (RDF), Linked Data, and REST. It does not address system lifecycle modeling.

As to modeling the system lifecycle, LML, introduced in the previous section and described further in the next section, positions itself as a lifecycle modeling language. In the area of automation systems, the Object-Process Methodology (OPM) [18] is a conceptual approach, language, and methodology for both the modeling and knowledge representation of such automation systems. Relating objects to processes, it offers graphical, hierarchical, interrelated diagrams, and on their basis a textual model can be generated. Yet OPM also positions itself as being domain-independent. Work regarding SysML-centric approaches includes Bajaj et al. [19], who describe MBSE++, their vision for model-based SysE across the system lifecycle with a unified Total System Model framework that utilizes SysML in conjunction with other models; they present a vision, challenges, and potential use cases exemplified with browser-based 2D visualizations on the Syndeia platform. For managing the lifecycles of the models themselves, Fisher et al. [20] focus on models as configuration items and repository support, mentioning Model Lifecycle Management (MLM) and MLM Systems (MLMS) to support a MLM processes for this context. They argue that Product Lifecycle Management (PLM) tools often lack good integration with MDE tools, while Application Lifecycle Management (ALM) are more SW-focused and lack system development capabilities. Here as well no lifecycle modeling language is addressed. Bork et al. [21] provide a survey comparing 11 (mostly visual) modeling language specifications, including LML, UML, BPMN, and OPM.

Regarding visualization, while typical VR use cases during engineering include modeling, simulation, and operation of the target system, we found no related work regarding the use of VR for visualizing the lifecycle of SysE or SE.

3 Background on the Lifecycle Modeling Language

LML [1] was selected as a basis modeling language for the SDLC for several reasons: 1) it's positioned specifically as a lifecycle modeling language, 2) it's maturity (since 2013 to version 1.4 in 2022), 3) it's relative simplicity and extensibility, 4) it's based on an ontology, 4) it's fundamental tenet that each entity has at least one corresponding visualization, and 5) the accessibility of a concrete format and sample model datasets for testing our solution concept.

LML uses common vocabulary to define modeling elements (e.g., entity, attribute, schedule, cost, relationship). Primary visual modeling constructs are the box (representing a part of the system) and the directed arrow (depicting relationships between elements, like "consists of," "derived from," or "costs"), to make it easy for various stakeholders to grasp. The LML ontology consists of 20 entities in total. The 12 primary (8 children) entities are Action, Artifact, Asset (Resource), Characteristic (Measure), Connection (Conduit, Logical), Cost, Decision, Input/Output, Location (Orbital, Physical, Virtual), Risk, Statement (Requirement), and Time. How the principal entities relate to each other and support traceability is shown in Fig. 1.

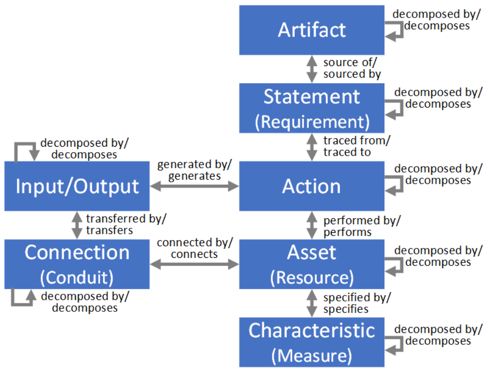


Fig. 1. Principal LML entities/relationships supporting traceability (adapted from [1]).

As to visualization, LML defines a logical construct using visualization models in four areas (primary entities) [22]:

- Functional models (Action, Input/Output),
- Physical models (Asset/resource, Connection),
- Documentation entities (Artifact, Statement/Requirement), and
- Parametric and program entities (Time, Risk, Cost, Characteristic/Measure, Location).

LML defines three key diagram types: Action (behavioral), Asset (physical), and Spider (traceability). While other diagrams are offered, these three express essential elements for systems engineering. While LML offers a metamodel, it lacks a visual metamodel, a coordinate system (for precisely specifying the height and width of modeling constructs), variables for specifying color or shape, yet it does offer horizontal and vertical position variables [21]. LML tools can also dynamically generate diagrams, which can help to ensure their veracity given a data-centric model. Since LML diagrams can be auto-generated by tooling, it moves SysE towards Data-Driven SysE (DDSE) rather than remaining focused on Model-Based SysE (MBSE) [2].

As to available mappings to other popular notations such as SysML and BPMN, a mapping of SysML to LML is addressed in Appendix A of the LML Specification [1]. As LML is extensible, Equation and Port were added to provide SysML visualizations. Any lifecycle modeling will involve modeling processes. While BPMN is popular and supports highly complex process models, version 2.0.2 has 116 elements. In contrast, LML retains a simple ontology and graphical notation. Mapping between BPMN and LML Action Diagrams is described by Stevie [23].

Innoslate by Spec Innovations [27] is a cloud tool offering model-based SysE supporting LML and other modeling languages. It integrates requirements, testing, V&V, system modeling, simulation, project management, and issue tracking. Via its web-based user interface, various LML and other (e.g., SysML) diagrams can be generated as shown as a dashboard of widgets in Fig. 2 and a single Action diagram in Fig. 3.

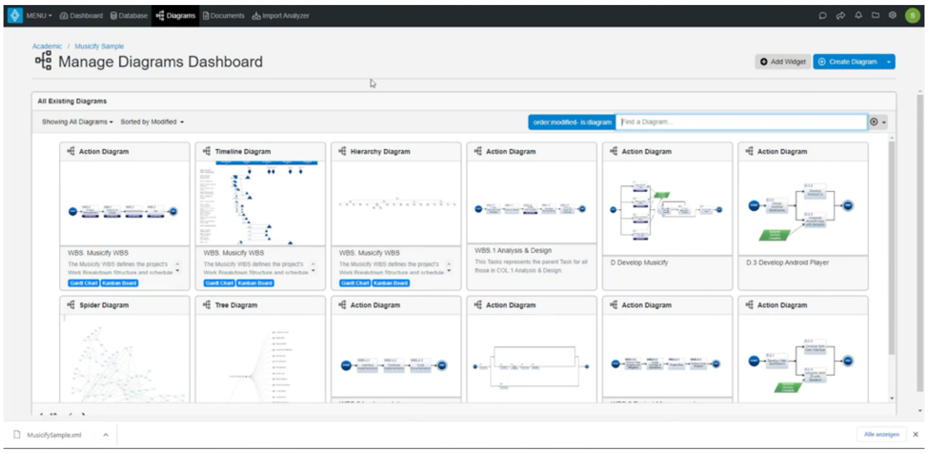


Fig. 2. Innoslate diagram dashboard screenshot for Musicify Sample project.

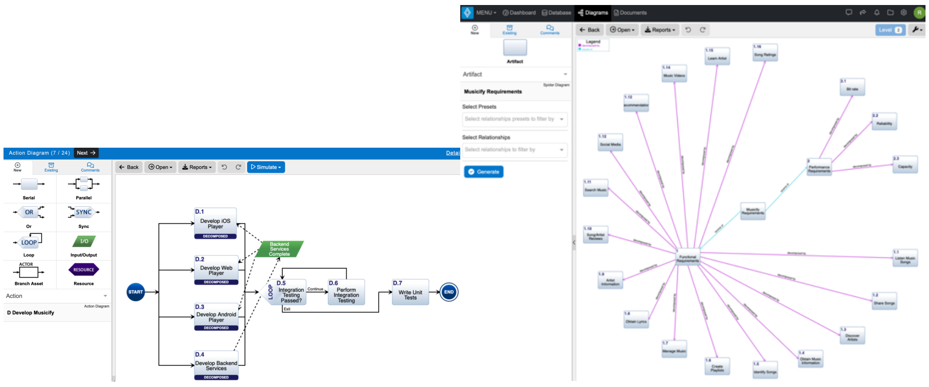

START
D.1 Develop iOS Player
D.2 Develop Web Player
D.3 Develop Android Player
D.4 Develop Backend Services
Backend Services Complete
D.5 Integration Testing Passed?
D.6 Perform Integration Testing
D.7 Write Unit Tests
END
LOOP
Functional Requirements

Fig. 3. LML Action (left) and Spider (right) diagram screenshots in edit mode in Innoslate.

4 Solution Concept

Our solution approach leverages VR for visualizing the various lifecycle and system models and their interrelationships simultaneously, mapped to a spatial structural model that can be immersively explored and experienced. The comprehensive integration of context as a first principle for lifecycles supports analysis and comprehension of the "big picture" for structurally and hierarchically complex and interconnected lifecycle elements, models, diagrams, and artifacts. It provides a 3D visualization for the entire lifecycle model, viewable and filterable from different perspectives by diverse stakeholders, such as system/software architects/engineers, requirements analysts, test engineers, operators, auditors, logisticians, and project/program managers.

4.1 Grounding in VR-Related Research

Our reasoning for applying VR in our solution concept is based on prior VR research in areas we view as related to modeling, analysis, and collaboration, some of which are highlighted here. In their systematic meta-analysis, Akpan & Shanker [24] showed VR and three-dimensional (3D) offer significant advantages in the area of Discrete Event Modeling (DES), including model development, analysis, and Verification and Validation (V&V), with the most consistent conclusions being the positive effects of 3D/VR model analysis and V&V. Of 23 articles examining 3D analysis, 95% concluded using 3D was more potent and lead to better analysis than 2D, e.g., when evaluating a model's behavior or performing a what-if analysis. They also found a consensus that 3D/VR can present results convincingly and understandably for decision-makers. 74% of 19 papers concluded that 3D/VR significantly improves the model development task (supporting teams and improving precision and clarity). To investigate VR's suitability for analytical tasks for an Information Architecture (IA), Narasimha et al. [25] used a card sorting collaboration experiment. They found that VR was at least as good as in-person card sorting, and that VR did not perform worse than video screen-share or VR; in fact, for certain variables VR was even better than both conventional and video-based conditions. Qualitative data evaluating awareness indicated that during collaborative interaction, participants were aware of their task, others, and their context, while collaborating similarly to an in-person setting. Additionally, the qualitative data showed evidence of positive views towards VR. The outcomes suggest that both a sense of presence and collaboration (equivalent to an in-person setting) is possible within VR. Another survey of Immersive Analytics (IA) by Fonnet & Prie [26] analyzed 177 papers. They found concurring evidence that for graph and spatial data analysis, IA provides benefits vs. non-IA when the scene complexity exceeds the 2D display, while for multi-dimensional data, the advantages are more task-dependent. They remark that while IA enables exploration of large-scale data worlds, context-aware navigation techniques are insufficiently exploited - although they are critical for users. We thus conclude that the immersive VR experience has significant potential for comprehensively depicting 3D models, supporting awareness, analysis, stakeholder inclusion, collaboration, and contextualization.

4.2 VR-SDLC Solution Concept

The contribution of our paper is a VR solution concept for the contextual visualization of the SE or SysE lifecycle models, including associated diagrams and heterogeneous models. Our solution concept is abstract, extensible, and leverages available lifecycle modeling construct providers (e.g., LML). We are not endorsing any particular language, scheme, methodology, product, or tool, but rather utilizing them to investigate the feasibility of utilizing VR for conveying SDLC with contextual aspects. The modeling constructs are typically extensible, should additional elements or diagrams be required. For complex systems and their lifecycle, VR's visual, immersive medium provides a collaborative opportunity for portraying lifecycle aspects in a contextually-relevant manner for diverse stakeholders

SysE, SE, and enterprise architecture (EA) are broad topics, yet having many facets and potentially overlapping commonalities. SysE and SE typically rely on and are executed in an organization such as an enterprise, which in turn has an EA. We thus developed various solutions concepts as shown in our solution map in Fig. 3. Our generalized VR Modeling Framework (VR-MF), described in [7], provides a domain-independent hypermodeling framework addressing key aspects for modeling in VR: visualization, navigation, interaction, and data retrieval. On this basis, VR-EA [7] provides specialized direct support and mapping for EA models in VR, including both ArchiMate as well as BPMN via VR-BPMN [11], while VR-ProcessMine [28] supports process mining in VR. VR-EAT [8] extends this to our enterprise repository integration solution, exemplified with Atlas integration, visualization of IT blueprints, and interaction capabilities. VR-EA+TCK [9] expands this further by integrating KMS and ECMS capabilities in VR. And VR-EvoEA+BP [10] adds VR support for animating enterprise evolution and (even non-BPMN) business processes. Our VR-based solutions specific to SE and SysE include: VR-SDLC (the focus of this paper, shown in blue), which could leverage our VR-V&V [29], VR-TestCoverage [30], VR-GitCity [31] and VR-Git [32] as well as our modeling solutions including VR-UML [13], VR-SysML [12], and VR-SysML+Traceability [33]. In supporting SDLC lifecycles with VR, we foresee future opportunities to leverage the aforementioned solution concepts in the enterprise and organizational space (e.g., knowledge, processes, infrastructure, architecture, artifacts) towards more comprehensive and holistic solutions in the (SE and SysE) engineering, development, and operations space as well.

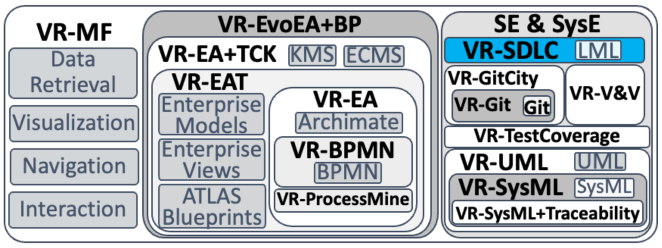


Fig. 4. VR solution concept map showing VR-SDLC (blue) in relation to our other concepts.

To achieve these objectives, our solution concept necessitates data integration and VR visualization, navigation, and interaction capabilities, addressed as follows:

Lifecycle data integration. As a representative tool and repository for lifecycle data integration, Innoslate provides diverse lifecycle-related data in a coherent repository and meta-model based on LML. The Innoslate cloud-based tool provides a 2D browser interface and offers sample LML projects that can be exported in its XML format.

Visualization. Since many possible relations between digital elements are possible, a spherical nexus was chosen to visualize all elements and relations for a project lifecycle. To provide some initial ordering, layering within the sphere is available as a grouping mechanism based on similar element types using the color assigned to that type, resulting in a sphere with colored layers (intra-layer element placement is random). A default (customizable) node color scheme is used. To assist with orientation and make interaction more intuitive by providing a context for what a model represents, labeled glass boxes readable from any angle contain a nexus based on some model. As 2D-based views and diagrams remain a primary form of SysE and SE documentation, they are visualized via 3D hyperplanes in proximity to its nexus for contextual support. In summary, hitherto intangible digital elements "or digital twins" are thus made visible and related to one another across the lifecycle spectrum.

Navigation. To support immersive navigation in VR while reducing the likelihood of potential VR sickness symptoms, two navigation modes are supported in the solution concept: the default uses gliding controls, enabling users to fly through the VR space and get an overview of the entire model from any angle they wish. Alternatively, teleporting permits a user to select a destination and be instantly placed there (i.e., moving the camera to that position), potentially reducing the probability of VR sickness when moving through a virtual space.

Interaction. Interaction in VR is supported primarily via the VR controllers and our VR-Tablet concept. Views consisting of diagrams are stacked as hyperplanes (with subdiagrams placed below), with corresponding objects highlighted in the Nexus or diagram with the other object is selected. Our VR-Tablet paradigm provides: interaction support, detailed information regarding a selected element, and can support browser-based multimedia content, browsing, filtering, and searching for nexus nodes.

5 Realization

The logical architecture of our VR-SDLC prototype realization is shown in Fig. 5, consists of the following modules: the Data Hub (with a GUI, an LML Adapter and Transformer, and a Database), and the Unity VR modules: General, Tablet, and LML-specific modules (ActionDiagrams, SpiderNexus, AssetDiagrams, and Connectors).

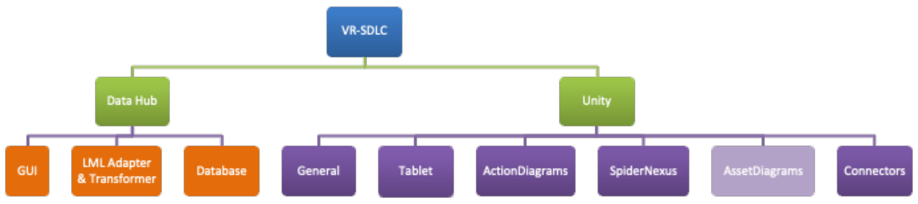


Fig. 5. VR-SDLC Logical Architecture.

The Data Hub GUI, realized with PythonQT, simplifies data import via drag-and-drop. Our Python-based adapter and transformer converts the XML exported from Innoslate. The main elements of the XML structure are: innoslate (the root element), schema, schemaClass, schemaRelation, xmlTargetRelation, database, entity (name, description, schemaClassId, numbering, creation/change timestamp, etc.), simulationData (Action Diagram structure), relationship (between two entities: sourceId, schemaRelationId, targetId). This is mapped to our JSON format, which is stored in the Neo4j graph database, which provides a flexible basis for supporting contextualization via its graph structure and Cypher Query Language. Fig. 6 exemplifies relations between Action Diagram elements viewed in the Neo4j database browser.

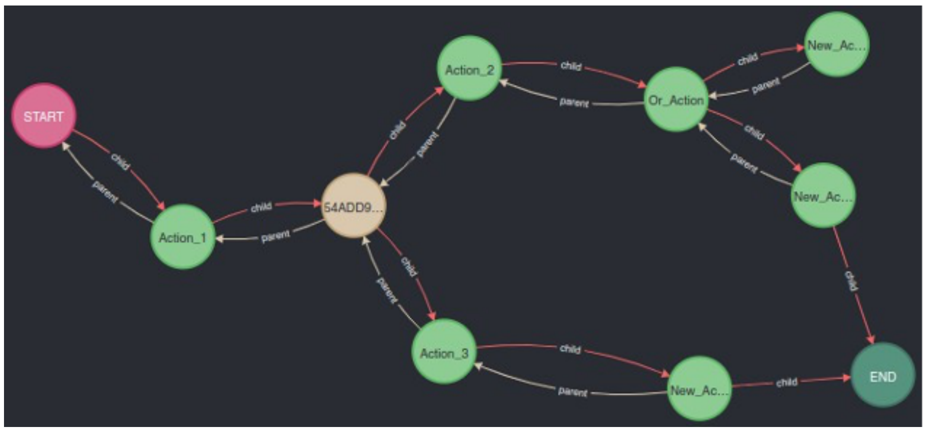


Fig. 6. Elements of an Action Diagram viewed in Neo4j Browser.

From the three primary diagram types specified in LML, our prototype realization currently supports the equivalent of Spider and Action diagrams in VR. The *Spider* diagram is mandatory for traceability, depicting how entities relate to one another. To realize the essence of what Spider diagrams convey in VR, we instead utilize a spherical nexus to visualize the relations between entities. The *Action* diagram - the only diagram type unique to LML - is used to express functional sequencing in a simplified way (vs. flow charts, UML/SysML Activity Diagrams, or BPMN) with a notation that supports comprehension by other stakeholders. Control node types consist of SERIAL, LOOP, PARALLEL, OR, START, END, and SNYC. We realized Action diagrams in VR as a 3D model of the diagram and its elements on a hyperplane. *Asset* diagrams provide a physical representation of design elements. They consist of Asset, Resource as a subclass of Asset, and Conduits. Due to time and resource constraints, however, our prototype does not yet support Asset diagrams. For visualization realization purposes they can be viewed as a simplified variant of the Action diagram, and thus we do not foresee any technical hindrances.

As to support for additional (non-primary) diagram types, our demonstrated support for many diagram types and notations in VR was enumerated with Fig. 4 in the prior section, and we thus can envision support for the wide variety of the optional and common diagram visualizations recommended for use with LML [22].

6 Evaluation

An empirical evaluation is out-of-scope for this paper but included in future work. For the evaluation of our solution concept with our prototype realization, we refer to the design science method and principles [34], in particular, a viable artifact, problem relevance, and design evaluation (utility, quality, efficacy). For this case study, we focus on supporting lifecycle comprehension, analysis, and contextualization. In particular, with regard to scenarios involving diverse project lifecycle instantiations (SysE and SE), heterogeneous side-by-side models (e.g., with SysML), and cross-project or program management aspects.

To verify support for differing lifecycles, the following LML sample project datasets from the Innoslate tool were used:

Autonomous Vehicle: a SysE sample representing the development of an autonomous vehicle for SAE International Level 5 full automation. It consists of 199 nodes and 886 links, including 39 Statements, 37 Requirements, 33 Tasks, 16 Actions, 11 Assets, 18 Dependencies, 12 Time Elements, and 8 Input-Outputs. It contains 11 Action Diagrams.

FireSAT Satellite: a space satellite system that performs early detection and warning for forest fires. This SysE sample is from the Applied Space Systems Engineering book. It consists of 347 nodes and 1270 links, including 4 Statements, 44 Requirements, 33 Tasks, 63 Actions, 90 Assets, 3 Resources, 7 Artifacts, 15 Test Cases, 11 Characteristics, 12 Time elements, 9 Input-Outputs, 17 Conduits, 3 Equations, and 18 Cost elements. It contains 27 Action Diagrams.

Musicify Streaming Service: a hypothetical music streaming service as a software-centric (SE) reference example for software engineers. It consists of 189 nodes and 722 links, including 3 Statements, 20 Requirements, 2 Artifacts, 32 Tasks, 36 Actions, 25 Assets, 10 Conduits, 13 Time Elements, 6 Cost Elements, 15 Dependencies, 1 Resource, and 22 Test Cases. It contains 19 Action Diagrams.

SPECTER Lunar Rover: a SysE sample inspired by NASA's Break the Ice Challenge for a system for excavation and delivery of icy regolith under extreme lunar conditions. It consists of 152 nodes and 558 links, including 1 Statement, 17 Requirements, 51 Tasks, 12 Actions, 13 Assets, 4 Conduits, 4 Issues, 3 Artifacts, 17 Time elements, etc. It contains 26 Action Diagrams.

6.1 Multi-Project and Program Management Analysis Support Scenario

Fig. 7 demonstrates the ability to load and compare multiple LML lifecycles simultaneously, each in a transparent cube containing a nexus of interconnected nodes representing its lifecycle elements. Each nexus is centered on a common axis, thus the larger the nexus (illusion of being shifted forward), the greater the number of nodes; in this case FireSAT (second from left) is the largest with 347 nodes. To provide order, each nexus is layered by colored node type. Thus, one can visually readily determine that FireSAT has more Assets (red nodes) than the other projects. The stats for each project are displayed as a type-colored legend near the nexus box. Furthermore, via the VR-Tablet a user can toggle show all diagrams across all projects (Fig. 8), which are placed on an axis in front of each nexus (to avoid collisions, they are currently spaced apart, but we intend to further optimize proximity placement). Thus, a user can quickly determine which, if any, diagrams are even available. Via the VR-Tablet, the available diagrams are scrollable by name, and can selected and shown.

Hence, cross-project coordination or program management analysis (e.g., quality assurance) can be readily undertaken, e.g., to consider lifecycle optimizations or improvements in efficiency or effectiveness measures. While at first glance it may seem cognitively complex, which in fact such lifecycles are, yet VR-SDLC offers an immersive overview of the reality that can then be adjusted for the interests of the stakeholder involved.

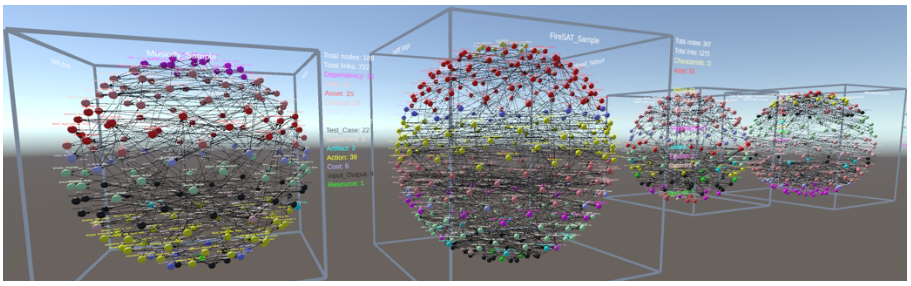


Fig. 7. Multiple LML projects loaded as VR-SDLC nexuses.

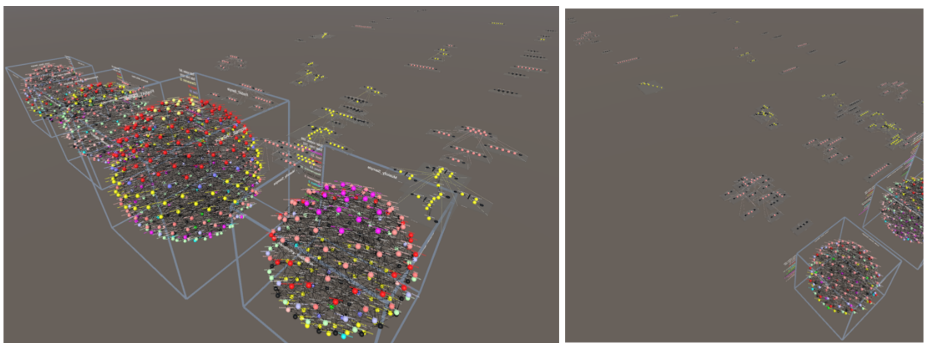


Fig. 8. VR-SDLC showing all diagrams available for multiple LML projects.

6.2 Contextualization Scenario

This scenario focuses on providing stakeholders with contextual support. Each nexus provides insight to the entire contextual information available to a given project, provided primarily via the links between nodes as shown in Fig. 9.

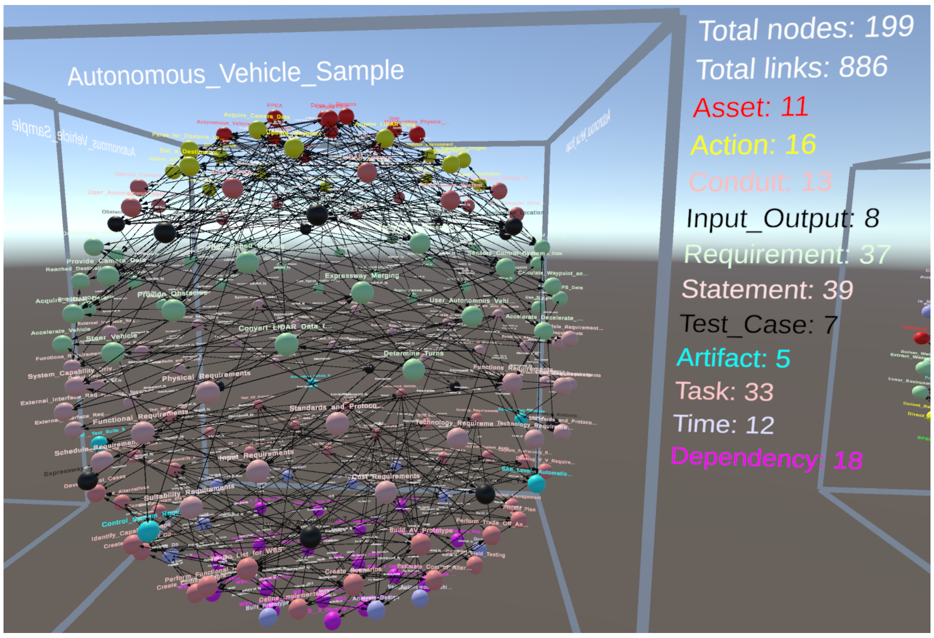
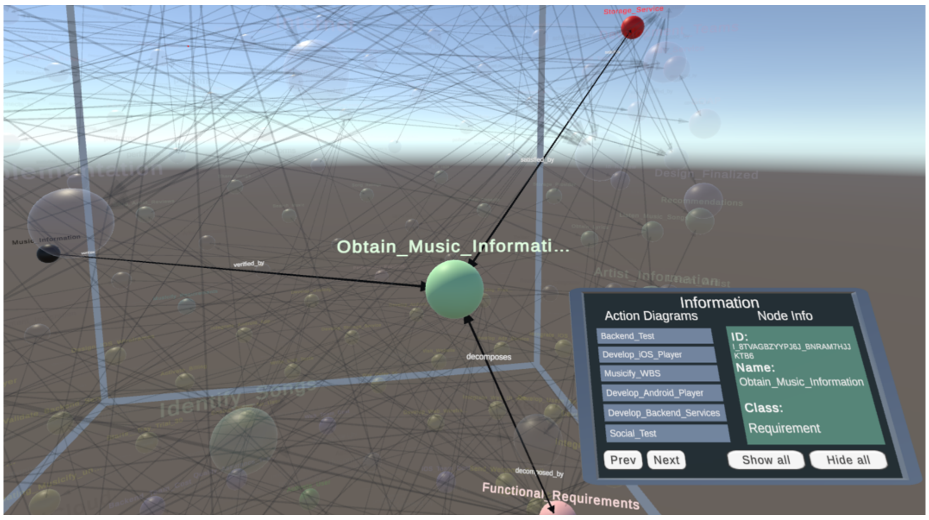


Fig. 9. VR-SDLC depicting LML Autonomous Vehicle Sample nexus.



Information

Action Diagrams
- Backend_Test
- Develop_iOS_Player
- Musicify_WBS
- Develop_Android_Player
- Develop_Backend_Services
- Social_Test

Node Info
ID: I_8TVAGBZYYPJ6J_BNRAM7HJJKTB6
Name: Obtain_Music_Information
Class: Requirement

Fig. 10. VR-SDLC depicting selection in LML Musicify Sample nexus.

For the context of a given element, if any element in a nexus is selected as shown in Fig. 10, only links to first-degree nexus neighbors remain colored, other nodes and links are ghosted. This highlights its direct context, while leaving hints at the overall context. Here, the node "Obtain Music Information" of type Requirement is selected, with the VR-Tablet providing additional details (colored by type) about data associated with the element. The first-degree neighbors and *relations* for this element are: *decomposes* the Statement Functional_Requirements (pink), is *satisfied_by* the Asset Storage_Service (red), and is *verified_by* the Test Case Music_Information (black).

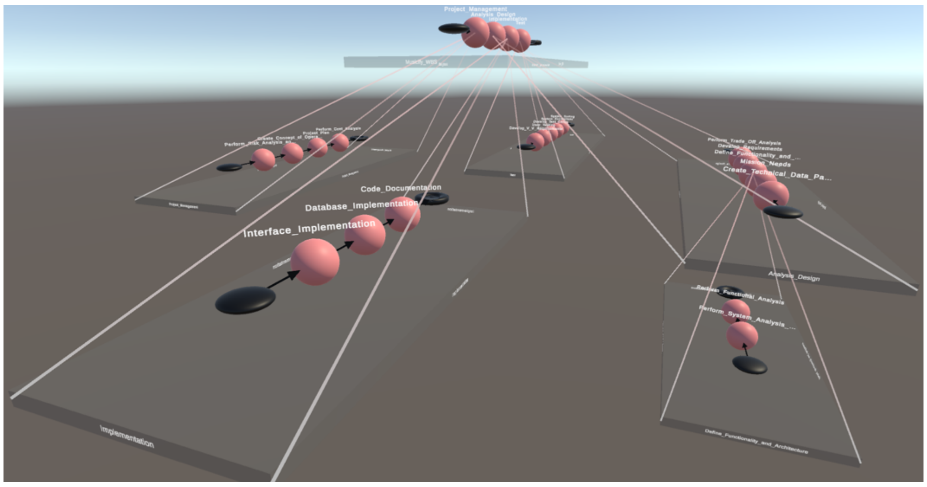
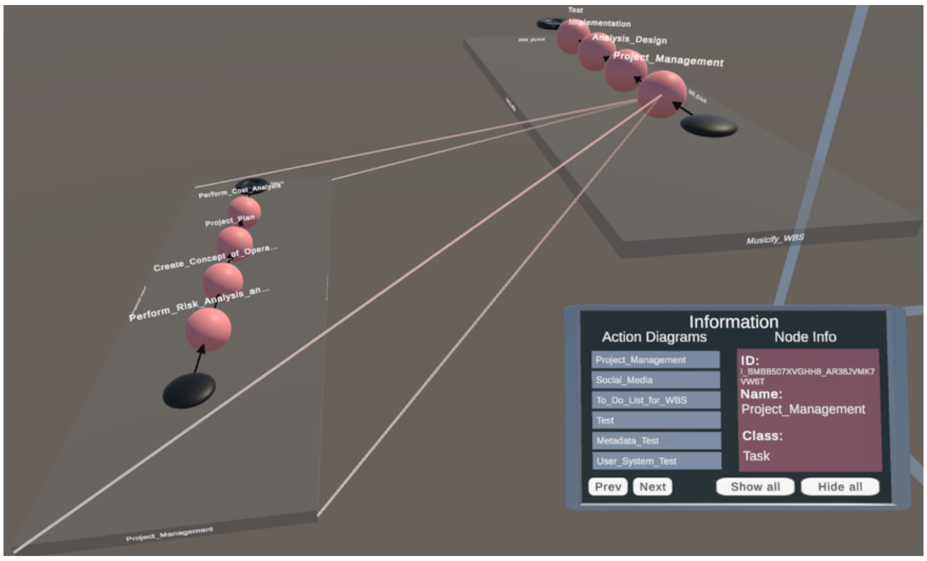


Fig. 11. VR-SDLC expanded contextual hierarchy of LML Action diagrams.



ID:
I_BMBB507XVGHH8_AR38JVMK7VW6T
Name:
Project_Management
Class:
Task

Fig. 12. VR-SDLC showing an LML Action subdiagram for the Project_Management Task.

For diagrams, context relative to other diagrams is provided if associations to sub- or super- diagrams exist (as illustrated in Fig. 11 for a set of diagrams and Fig. 12 for a single Task). Second, selecting an element in either the nexus or diagram, only the links in the nexus to first-degree neighbors remain colored, all other nexus nodes and links are ghosted, and only the diagrams associated with that node are shown. This can be seen in Fig. 13, where the Task Musicify WBS was selected.

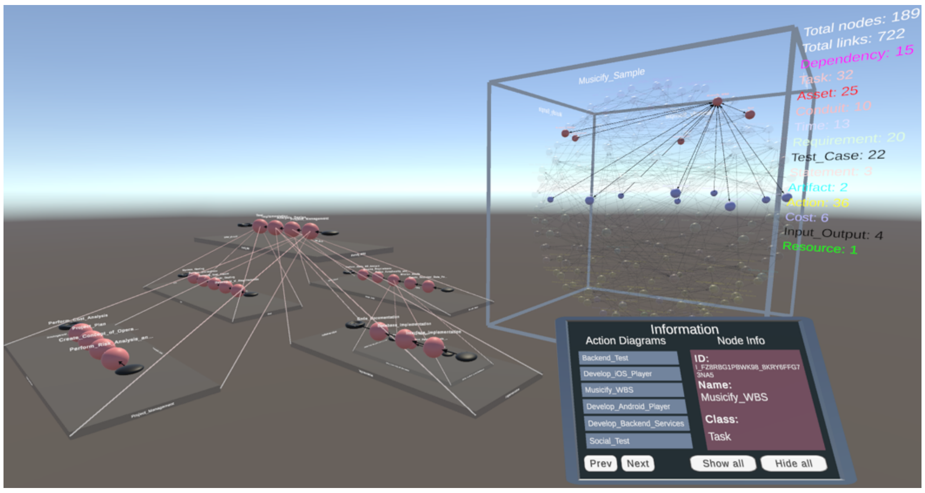
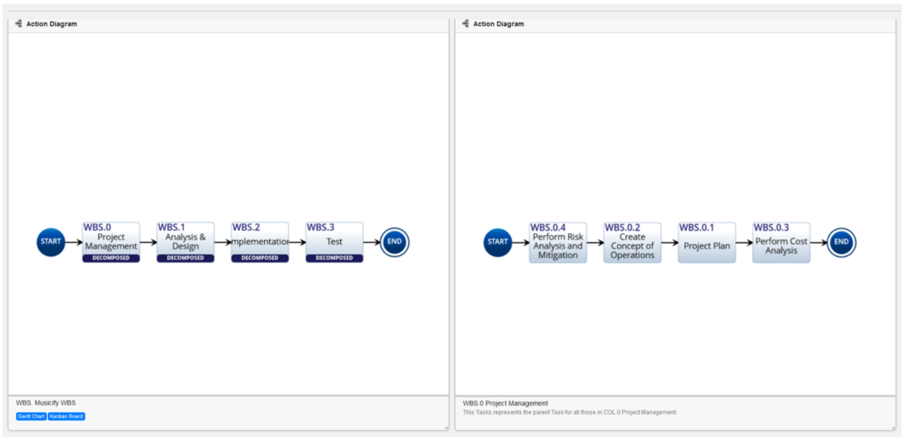


Fig. 13. VR-SDLC Contextualization of Actions in Nexus and multiple LML Diagrams.

In contrast, in a browser using 2D the diagrams, this contextual information can be difficult to attain. For instance, in the Innoslate tool, when one double-clicks one element, then the view typically changes to the subdiagram. One can place diagrams side-by-side (see Fig. 14), but it is not visually obvious (but rather possible via numbering) to extract how the elements and diagrams possibly relate.


Action Diagram
START
WBS.0 Project Management DECOMPOSED
WBS.1 Analysis & Design DECOMPOSED
WBS.2 mplementatior DECOMPOSED
WBS.3 Test DECOMPOSED
END
Gantt Chart
Kanban Board
Action Diagram
START
WBS.0.4 Perform Risk Analysis and Mitigation
WBS.0.2 Create Concept of Operations
WBS.0.1 Project Plan
WBS.0.3 Perform Cost Analysis
END

WBS. Musicify WBS

WBS.0 Project Management

This Tasks represents the parent Task for all those in COL.0 Project Management.

Fig. 14. LML Action Diagrams placed side-by-side as widgets in Innoslate.

6.3 Heterogeneous Hypermodeling Support Scenario

Immersive heterogeneous multi-model analysis is supported by loading multiple models in VR as shown in Fig. 15, where SysML diagrams (as hyperplanes) are loaded via VR-SysML [12] near the LML models. In previous work, we have demonstrated loading heterogenous models in VR, such as our VR-UML [13], VR-EA for ArchiMate [7], and VR-BPMN [11] solutions models. The unlimited space in VR offers the ability to be cognizant of the various models available and how they relate, and can offer new insights into potential missing, forgotten, or duplicated aspects and could support a more comprehensive traceability.

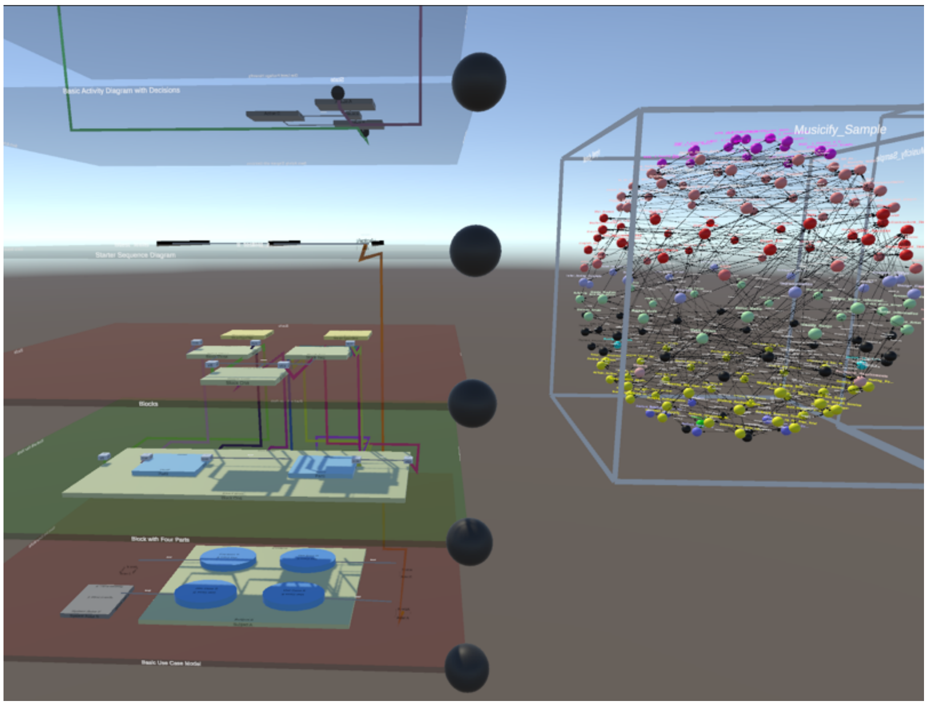


Fig. 15. Example heterogeneous multi-model visualization and analysis capability showing SysML diagrams (as hyperplanes) near LML models.

Furthermore, integration with our EA-centric VR-EA+TCK [9] is readily feasible, and would enable a much more comprehensive and holistic insight for stakeholders, taking into account the entire organization (as the enterprise) with all its related elements and systems, such as KMS and ECMS. Furthermore, a time-based lifecycle or target system evolution with a process-mining view as demonstrated in VR-EvoEA+BP [10] could be applied at the project-level to the SDLC lifecycle as well. Integration with VR-BPMN [11] models is also feasible, if such process models were available to the lifecycle or if the available mapping of LML to BPMN were desired. We thus see further opportunities for multi-model integration with VR-SDLC.

7 Conclusion

Current modeling notations for lifecycle and system/software models are available, yet are typically 2D diagram-focused and run into inherent visual limitations, lacking the ability to comprehensively visualize the lifecycle, to do this contextually, and to visually integrate the lifecycle with its target models. VR-SDLC contributes a solution concept for an immersive visualization experience of SysE and SE lifecycle models, supporting diverse stakeholders with context-enhanced insights into the lifecycle. Via its nexus-based approach, VR-SDLC comprehensively portrays lifecycle model elements, relations, and diagrams while enhancing them with context. The solution concept was demonstrated with our prototype realization that showed support for diverse LML model lifecycles. Multiple lifecycle projects can be portrayed concurrently to support inter-project or program portfolio analysis and optimizations. Its hypermodeling capability was demonstrated with VR-SysML, showcasing how both a target model in SysML and its lifecycle in LML can be portrayed concurrently in VR. The VR-SDLC solution concept is abstract and flexible, and alternative modeling languages, data formats or integrations, and additional modeling elements or diagram types could be readily supported. VR-SDLC could be used to enhance or extend existing very capable 2D lifecycle and modeling tools such as Innoslate. The benefits of our solution approach include comprehensive lifecycle model visualization, supporting comprehension, analysis, and contextualization. This includes heterogeneous side-by-side lifecycle models lifecycle with the target models, enabling the tailoring, traceability, and verification of lifecycle activities to the target model (e.g., UML/SysML) situation. Furthermore, the ability to load various lifecycle models in VR simultaneously supports cross-project or program management analysis. By portraying the lifecycle contextually and comprehensively, more inclusive and intuitive stakeholder collaboration and contextual insights are supported. Future work includes realizing support for Asset Diagrams, model modification in VR, OSLC support, automated model verification, and a comprehensive empirical study.

Acknowledgements. The authors would like to thank Sasche Roelofs and Jonas Kling for their assistance with the design, implementation, evaluation, and figures. Further, we would like to thank Spec Innovations for providing academic access to their Innoslate tool and the accompanying sample data sets.

References

1. Lifecycle Modeling Language Steering Committee: Lifecycle Modeling Language (LML) Specification, Version 1.4 (October, 2022). https://www.lifecyclemodeling.org
2. Vaneman, W. K., Sellers, J. J., Dam, S. H.: Essential LML: Lifecycle Modeling Language (LML): a Thinking Tool for Capturing, Connecting and Communicating Complex Systems. SPEC Innovations (2018)
3. Object Management Group: BPMN Specification 2.0.2 (2014)
4. Object Management Group: Unified modeling language version 2.5.1 (2017)

5. Object Management Group: Systems modeling language version 1.6 (2019)
6. International Council on Systems Engineering (INCOSE): Systems Engineering Vision 2025. INCOSE, (2014). Accessed 20240319: https://web.archive.org/web/20190308160903/http://www.incose.org/docs/default-source/aboutse/se-vision-2025.pdf
7. Oberhauser, R., Pogolski, C.: VR-EA: Virtual Reality Visualization of Enterprise Architecture Models with ArchiMate and BPMN. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2019. LNBIP, vol 356. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24854-3_11
8. Oberhauser, R., Sousa, P., Michel, F.: VR-EAT: Visualization of Enterprise Architecture Tool Diagrams in Virtual Reality. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2020. LNBIP, vol 391. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52306-0_14
9. Oberhauser, R., Baehre, M., Sousa, P.: VR-EA+TCK: Visualizing Enterprise Architecture, Content, and Knowledge in Virtual Reality. In: Shishkov, B. (eds) Business Modeling and Software Design (BMSD 2022), pp. 122-140. LNBIP, vol 453. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-11510-3_8
10. Oberhauser, R., Baehre, M., Sousa, P.: VR-EvoEA+BP: Using Virtual Reality to Visualize Enterprise Context Dynamics Related to Enterprise Evolution and Business Processes. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2023. LNBIP, vol 483. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-36757-1_7
11. Oberhauser, R., Pogolski, C., Matic, A.: VR-BPMN: Visualizing BPMN Models in Virtual Reality. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2018. LNBIP, vol 319. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94214-8_6
12. Oberhauser, R.: VR-SysML: SysML model visualization and immersion in virtual reality. In: Proceedings of the International Conference of Modern Systems Engineering Solutions (MODERN SYSTEMS 2022), pp. 61-66. IARIA, (2022)
13. Oberhauser, R.: VR-UML: The Unified Modeling Language in Virtual Reality – An Immersive Modeling Experience. In: Shishkov, B. (eds) Business Modeling and Software Design. BMSD 2021. LNBIP, vol 422. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79976-2_3
14. ISO/IEC/IEEE: ISO/IEC/IEEE 15288:2023 - Systems and software engineering — System life cycle processes (2023)
15. ISO/IEC/IEEE: ISO/IEC/IEEE 12207:2017 - Systems and software engineering — Software life cycle processes (2017)
16. Dori, D.: Model-based standards authoring: ISO 15288 as a case in point. Systems Engineering, 27(2), pp. 302-314 (2023). https://doi.org/10.1002/sys.21721
17. OASIS Standard: OSLC Core Version 3.0. Part 1: Overview. (2021)
18. ISO: ISO/PAS 19450:2024 Automation systems and integration – Object- Process Methodology (2024)
19. Bajaj, M., Zwemer, D., Yntema, R., Phung, A., Kumar, A., Dwivedi, A., Waikar, M.: MBSE++—Foundations for Extended Model-Based Systems Engineering Across System Lifecycle. In INCOSE International Symposium, Vol. 26, No. 1, pp. 2429-2445 (2016)
20. Fisher, A., Nolan, M., Friedenthal, S., Loeffler, M., Sampson, M., Bajaj, M., VanZandt, L., Hovey, K., Palmer, J., Hart, L.: Model Lifecycle Management for MBSE. INCOSE International Symposium, 24: 207-229 (2014). https://doi.org/10.1002/j.2334-5837.2014.tb03145.x
21. Bork, D., Karagiannis, D., Pittl, B.: A survey of modeling language specification techniques. Information Systems, 87, 101425 (2020). https://doi.org/10.1016/j.is.2019.101425

22. Vaneman, W. K.: Enhancing model-based systems engineering with the Lifecycle Modeling Language. 2016 Annual IEEE Systems Conference (SysCon), pp. 1-7. IEEE (2016). doi: 10.1109/SYSCON.2016.7490581.
23. Stevie, L.: Business Process Model Notation and the Lifecycle Modeling Language (2023). Accessed 20240319: https://www.lifecyclemodeling.org/updates/bpmn-and-lml
24. Akpan, I. J., Shanker, M.: The confirmed realities and myths about the benefits and costs of 3D visualization and virtual reality in discrete event modeling and simulation: A descriptive meta-analysis of evidence from research and practice. Computers & Industrial Engineering, 112, 197-211 (2017)
25. Narasimha, S., Dixon, E., Bertrand, J. W., Madathil, K. C.: An empirical study to investigate the efficacy of collaborative immersive virtual reality systems for designing information architecture of software systems. Applied ergonomics, 80, 175-186 (2019)
26. Fonnet, A., & Prie, Y.: Survey of immersive analytics. IEEE transactions on visualization and computer graphics, 27(3), 2101-2122 (2019)
27. Innoslate, https://specinnovations.com/innoslate
28. Oberhauser, R.: VR-ProcessMine: Immersive Process Mining Visualization and Analysis in Virtual Reality. In: The Fourteenth International Conference on Information, Process, and Knowledge Management (eKNOW 2022), pp. 29-36. IARIA (2022)
29. Oberhauser, R.: VR-V&V: Immersive Verification and Validation Support for Traceability Exemplified with ReqIF, ArchiMate, and Test Coverage. International Journal on Advances in Systems and Measurements, 16(3 & 4), 103-115 (2023)
30. Oberhauser, R.: VR-TestCoverage: Test Coverage Visualization and Immersion in Virtual Reality. In Proceedings of the The Fourteenth International Conference on Advances in System Testing and Validation Lifecycle (VALID 2022), pp. 1-6. IARIA (2022)
31. Oberhauser, R.: VR-GitCity: Immersively Visualizing Git Repository Evolution Using a City Metaphor in Virtual Reality. International Journal on Advances in Software, 16(3 & 4), 141-150 (2023)
32. Oberhauser, R.: VR-Git: Git Repository Visualization and Immersion in Virtual Reality. In Proceedings of the the Seventeenth International Conference on Software Engineering Advances (ICSEA 2022), pp. 9-14. IARIA (2022)
33. Oberhauser, R.: VR-SysML+Traceability: Immersive Requirements Traceability and Test Traceability with SysML to Support Verification and Validation in Virtual Reality. International Journal on Advances in Software, 16(1 & 2), 23-35 (2023)
34. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly, 28(1), pp. 75-105 (2004)