RASAECO: Requirements Analysis of Software for the AECO Industry

Marko Ristin*, Dag Fjeld Edvardsen†, Hans Wernher van de Venn*

*Zurich University of Applied Sciences (ZHAW), {rist, vhns}@zhaw.ch
†Catenda AS, dag.fjeld.edvardsen@catenda.no

***Abstract*—Digitalization is forging its path in the architecture, construction, engineering, operation (AECO) industry. This trend demands not only solutions for data governance but also sophisticated cyber-physical systems with a high variety of stakeholder background and very complex requirements. Existing approaches to general requirements engineering ignore the context of the AECO industry. This makes it harder for the software engineers usually lacking the knowledge of the industry context to elicit, analyze and structure the requirements and to effectively communicate with AECO professionals. To live up to that task, we present an approach and a tool for collecting AECO-specific software requirements with the aim to foster reuse and leverage domain knowledge. We introduce a common scenario space, propose a novel choice of an ubiquitous language well-suited for this particular industry and develop a systematic way to refine the scenario ontologies based on the exploration of the scenario space. The viability of our approach is demonstrated on an ontology of 20 practical scenarios from a large project aiming to develop a digital twin of a construction site.**

I. Introduction

Projects, big and small alike, need to anticipate user expectations and scope in form of requirements. This is especially true for large software projects in the architecture-construction-engineering-operation (AECO) industry: they tend to be heavily invested in not only software complexity, but also complex business logic, industry practices, legal rules and a myriad of stakeholders with non-aligned interests and different technical backgrounds [1]. This complexity makes the understanding between software engineers (programmers, software architects *etc.*) and non-software engineers (construction engineers, business people, lawyers *etc.*) paramount for project success.

General software projects have both the problem and the solution space open, *i.e.*, the engineers developing the system need to capture the stakeholder requirements (the problem space) as well as come up with the system requirements (the solution space) [2]. In contrast, and analogous to fields such as smart manufacturing [3], the requirements engineering of the software for AECO industry need to consider many procedures, best practices and regulations already established in the industry. The stakeholder requirements and thus the problem space is given to a large extent, as opposed to general projects. Consequently, the requirements engineering shifts to the solution space to identify system requirements in the specific context of the industry.

To that end, we tailor a scenario-based approach [4] to requirements analysis suited for large AECO software projects addressing the following questions about the solution space:

- What do the scenarios share *in the context of AECO*?
- What *language* should we use to make the scenarios more formal and readable by both requirements engineers *and* AECO professionals?
- How can we refine the scenarios *systematically*?
- How can we *reuse* the results *between* the projects?

For example, consider the case of risk management on a construction site. There exist standardized procedures how risks should be identified, tracked and managed. There are predefined actions that need to be undertaken in specific phases of the building lifecycle (*e.g.*, risks that are considered during the planning phase and then tracked during the construction phase) and at different levels of abstraction (*e.g.*, person-specific risks and site-specific risks, respectively). The nomenclature and the data structures such as Building Information Modeling (BIM [5]) are widely used in the industry and even mandatory in many publicly-funded construction projects.

Mapping such procedures to a software system as "black boxes" ignoring the AECO context is usually tedious and wasteful. The terms for both functional and non-functional requirements, and their grouping and organization need to be constantly reinvented from project to project. This is particularly hard if the requirements engineer lacks the domain expertise, which can often be assumed to be the case in practice.

In contrast, we propose a method to analyze the requirements for software in the AECO industry. We argue that it is much easier if the structure of the solution space is already preset and needs to be "filled" rather then re-invented each time. We developed a conceptual framework to capture the scenarios in a structured manner with the industry context in mind, including not only the domain of data governance but also cyber-physical systems. Our contribution is four-fold:

- We introduce a novel *scenario space* to highlight the common dimensions of the AECO procedures.
- Instead of common languages, such as UML [6], we explore a novel choice and use Building Information Model (BIM) and its open standard Industry Foundation Classes (IFC [5]) as our *main modelling language*.
- The relations between scenarios can be thought of as

an ontology. We propose a novel systematic approach to refining the AECO scenario ontologies based on our predefined and uniform space. The wasteful random explorations are minimized and reusable parts outlive the individual projects.
- Finally, as appropriate tools are crucial for requirements analysis [7], we provide *a software tool* to analyze and visualize the resulting requirements [8].

We call our approach "Requirements Analysis of Software for the AECO Industry" or RASAECO, for short.

In Section II, we summarize the prior art and motivate our approach. Sections III and IV describe the development environment of the approach, a large AECO software project, and explain the approach in detail accompanied by motivating examples, respectively. section V explains how an open source tool developed in this project helps analyze the requirements in a scalable manner. Section VI describes how we applied the approach on gathering and analyzing the requirements for the project. Section VII investigates how well the approach generalizes to other projects in AECO industry. We conclude the work in Section VIII and examine the limits of our approach and future work.

II. Motivation and Related Work

Requirements engineering is a well-established branch of software engineering and many other industries, and is also finding its ways into AECO [9].

As the rich body of literature suggests, domain-specific requirements engineering of software is necessary and beneficial [7]. Yet there is a limited number of works about how to approach the requirements engineering for AECO-related software. There exist collections of AECO use cases (e.g., [10]), but they lack a uniform solution space. Therefore the cases are narrow and disconnected, which does not promote clarity.

There are studies on the application of existing *conventional* requirements engineering techniques on the data governance software for the AECO industry [11], [12], [13], [14], [15], [16]. Their narrow focus and the disregard of the specifics of the industry make them ill-suited for generalization to wider range of AECO systems (*e.g.*, cyber-physical ones).

Process models based on the languages (such as IDM [17] and BPMN [6]) can be included in the requirements (*e.g.*, if a full specification is required like in [6]), but their focus on processes is too elaborate for the general requirements analysis [17].

Concrete scenarios for cyber-physical systems in AECO were explored in [18], [19]. Our work does not present concrete scenarios, but a framework how to structure and investigate such scenarios, including the unstructured scenarios given in [18], [19].

The general tools for specifying the requirements are abundant. It has been widely recognized that a successful project needs key requirements [2] to support finding a feasible solution in a targeted manner, where structure [20], semantic annotations [21], requirements templates and patterns [22], and ontologies [23] are all beneficial. In this vein, we provide a tool for structuring *AECO-specific* requirements annotating them with *AECO-specific* semantic tags and refining their ontologies in a systematic *AECO-specific* way. To the best of our knowledge, it is a first of its kind in this particular industry. In contrast to sophisticated annotation tools such as ReqPat [24], our annotations are light-weight and provide only guidance to the reader as modelling at the text level proved too high a barrier for AECO professionals (see Section IV-C).

Visualization of requirements is important [25]. Our work visualizes the requirements in spaces similar to the existing approaches in requirements engineering (*e.g.* [26]).

There is also a fruitful parallel research in context of smart manufacturing to specify a solution space (termed "reference architecture") to structure the discussions, and organize standards and implementations (RAMI4.0 [3] and many others). Our approach is strongly influenced by these trends and builds a scenario space for AECO following this pioneering work.

III. The BIMprove Project

The current approach was incrementally developed during the BIMprove Project [27], an initiative funded by the European Union's Horizon 2020 Research and Innovation Program with focus on building a low-carbon, climate-resilient future. The project started in September 2020 and aims to develop a dynamic digital system for the construction industry, increase productivity, cut costs and improve the working conditions. The system should deal both with static ("as-planned") as well as dynamic real-time data ("as-built" or "as-observed"). The participants include 12 partners (2 universities, 3 research centers, 1 non-profit organization and 6 industrial partners). The project will take 3 years with a total budget of 5.6 Mio. euros. The team involves more than 40 people including 6 software engineers, 15 experts in the AECO domain, 13 experts in robotics and mechanical engineering, 9 user experience experts, 2 experts in standardization *etc*. The authors of this work also participate in the project: two authors with a decade-long experience in the AECO industry and one author with a decade-long experience in software engineering.

The project spans a large and variate scope: deviation between "as-built" *versus* "as-planned" in many different settings (including augmented and virtual reality), enforcing cleanliness on the site, planning, tracking, and preventing risks, giving guidance on the site (*e.g.*, for deliveries or work), *etc.* The project is expected to result in an experimental system ready for further development and deployment.

IV. Our Approach: RASAECO

A. System Scenarios

We propose to analyze requirements in software projects within the AECO industry based on **scenarios**. Scenarios are a popular tool in requirements engineering [4] in which the requirements are gathered through reasoning about typical (real or imagined) user activities and system-user interactions. In our context, a scenario is an abstract formal or semi-formal descriptions of an AECO process at the level of a system or a model. It is not a narrative, but a structured document written

by requirements engineers giving a process description from the perspective of different protagonists in the construction and operation process of a building.

As our domain entails cyber-physical elements, we explicitly shift the focus from the *user* to the *system* as recognized by [18]. In such systems the real physical world increasingly interacts with its virtual representation, making the end-user interactions equally relevant as interactions between the different system modules (and in advanced stages of a project, inter-system interactions). Similar to "system stories" in [28], we used the explicit term **system scenarios** whenever there was potential for misunderstandings [1].

B. Scenario Space

Requirements engineers, especially the novices without the AECO experience, can be at loss how to compile scenarios [29]. The boundless scenario space provides little reuse between and within projects [22] and the requirements engineers need to re-define it for each problem at hand. Furthermore, an open scenario space also puts the burden on the reader as there might be no inter-project or even inter-scenario consistency thus forcing the reader to repeatedly infer it.

We therefore propose to set up an AECO-specific **scenario space** as a representation of any action taking place in the dimensions of the AECO industry. This should make our scenarios uniform and easier to both write and read by dissecting the requirements along the predefined dimensions. We propose the following three dimensions:

1) Aspects,
2) Phases in the building lifecycle, and
3) Hierarchy levels of detail.

The scenario space is defined in such a way that any AECO process can be mapped within these three dimensions. A system fulfilling the complete scenario space is thus capable of processing upcoming real-world events (correction of the building structure, rescheduling, security measures *etc.*) within the system. Consequently, the system design and its implementation are easier to model and can be explicitly traced back to the scenario space. Similar to software modules in a computer program, an individual scenario occupies only a portion of the scenario space by design, and we explicitly do not strive for scenarios which cover the whole space, as this would be both impractical to write and overwhelming to read. Instead, we disassemble the more complex scenarios into less complex ones, allow concrete scenarios to specify the details of the more abstract ones, and relate the scenarios among themselves (see Section IV-D). For example, a complex AECO procedure can be split into a series of steps, where each scenario represents a single step, and the bird-view of the procedure is encapsulated in a bundling scenario. The final 3D puzzle of all the scenarios is expected to fully populate the relevant scenario subspace of the project.

[1]The term "use case" or "system use case" would have been perhaps more precise and more in line with the existing literature [2]. However, we discarded it since it sowed too much confusion with the AECO professionals in our case, where the term "scenario" or "system scenario" was much better understood and used. This matches the colloquial usage of the term that was also employed in some works, *e.g.* [18].

a) Truck Guidance

"External truck drivers arrive at the construction site to deliver cargo."

As-planned	The deliveries are specified as tasks.
As-observed	The driver's device tracks the GPS location.
Divergence	The device guides the driver to the delivery location.
Scheduling	The unmet deadlines are pushed as topics.
Analytics	The statistics of delivery delays are reported.

b) Risk Management

"Different risks are planned and tracked on the site."

As-planned	The risk manager manages the risks.
As-observed	The risk manager orders focus spots for the recording.
Divergence	The preventive resource visually inspects the recordings.
Scheduling	The risk manager inspects the risk linked to the tasks.
Safety	The risk manager marks the dangerous tasks.
Analytics	The system tracks the escalated risks per category.

c) Cost Tracking

"The planned costs are tracked against the expenditures."

Cost	Planned costs and expenditures are related to tasks.
Analytics	Over-budget tasks are reported.

TABLE I: Summarized aspects of three scenarios from [31]. The aspects provide support for the requirements engineer during writing as well as for AECO experts during reading and reviewing, in particular given their familiarity with them through the "higher" BIM dimensions [30]. In contrast, conventional requirements analysis would have grouped them in ways unintuitive to AECO experts such as functional, non-functional and system requirements, while the requirements engineers without the industrial experience would probably miss some of the important AECO-specific aspects (see also Section VI).

Aspects address a concern of a multi-layered scenario. Unlike classical requirements engineering, which usually groups requirements into functional, non-functional and technical [12], and do not convey any domain-specific meaning, our aspects are specially tailored for the AECO industry, influenced by "higher" (5D, 6D *etc.*) BIM dimensions [30]. We identify the following aspects:

- *As-planned*: models of plans and planned activities (both in space and over time),
- *As-observed*: models of observations (both in space and over time),
- *Divergence*: difference between the as-planned and the observed data,
- *Scheduling*: logical order of activities and how it affects overall schedule,
- *Cost*: financial background of the scenario,
- *Safety*: safety regulations of workers and other actors (legal and non-legal) concerning the scenario and how the system should accommodate them,
- *Analytics*: expected analytics, both technical (*e.g.*, pipe routings, static and dynamic re-calculations) and business-related (*e.g.*, key performance indicators).

Fig. I gives reduced examples of the aspects of a real-world AECO scenario from our project (see Section III).

The risk manager inputs the initial set of risks already known during the planning.[planning]

Both the risk manager and the preventive resource can insert new and change existing risks accordingly.[construction]

Fig. 1: Example of a requirement from Table I b) "Risk Management" marked with phases in the building lifecycle. In this example, it was initially not clear that the second sentence referred to the "construction" instead of the "planning" phase. The conventional approach to requirements analysis would ignore the phases or refer to them implicitly. Our approach provides a standardized way of how to consider the phases in the building life cycle, thus making it easier both for writers and readers to deal with them explicitly.

A large project usually spans multiple **phases of the building lifecycle**. Consequently, the requirements in a scenario follow the lifecycle, but often cannot be structured in a linear manner with the respect to time as other groupings might make more sense (*e.g.*, grouping requirements by functionality). The reader should thus always be put in context with clear references to phases in case of ambiguities.

How the phases should be split depends on many factors such as organizational and contractual criteria. We modeled the phases based on business practices of our industrial partners in the project [27], but alternatives such as RIBA [32] are equally plausible. The phases thus depend on a concrete project, company, national and international environment. We propose the following ones:

- Planning,
- Construction,
- Operation,
- Renovation, and
- Demolition.

The relevant parts of the scenario document should be appropriately marked with the corresponding phase. Figure 1 shows an example of such a marked requirement.

Complex scenarios require the analysis of multiple **hierarchy levels of detail** in parallel [33], from a device to the network of companies, from component-level to system-level, with multiple levels interacting with each other. To that end, we propose the following hierarchy inspired by RAMI4.0 [3] to capture and situate requirements in multiple abstraction levels:

- Device or Person (*e.g.*, a shovel, or an electrician),
- Machine or Crew (*e.g.*, excavator, or a team of electricians),
- Site unit (*e.g.*, a certain zone on the construction site),
- Site,
- Site office,
- Company, and
- Network of companies.

The truck driver's device tracks the GPS location, but does not send it to the system.[device]

The location is only used to navigate the truck.[machine]

Fig. 2: Example of a requirement from Table Ia) "Truck Guidance" marked to highlight the level of detail. The marking enhances the reading by putting the extra accent. The reader is made aware that the location is only relevant for the machine (truck), but not the system. The predefined abstraction levels allow for consistent markings throughout the project while arbitrary markings usually employed in conventional requirements engineering would differ from scenario to scenario. This makes it easier for the writer using our approach to label the text as no custom levels need to be invented, but can be readily picked off-the-shelf. At the same time, the consistency of the levels throughout the project makes the reading also easier.

The hierarchy follows the organization of the AECO industry and how it is decomposed into units: a device is a hand tool or a part of a bigger system (*e.g.*, a smartphone as a part of the communication system), a machine is a more complex system and can contain different devices (*e.g.*, an excavator containing a shovel, hydraulic cylinders *etc.*), a site unit is responsible for machines and teams *etc.* The levels should therefore allow for more straightforward mapping from the AECO procedures to software requirements.[2] The levels of detail in the hierarchy are marked in the document analogous to phases. Figure 2 presents an example marking.

Employing three dimensions allows for a nice metaphorical visualization of the scenario space as a volumetric (*i.e.*, individual scenarios are represented as a set of cubes in that 3D space). Figure 3 demonstrates such a visualization of an example scenario. This visualization is especially practical when multiple scenarios are displayed, *e.g.*, in a list accompanied by thumbnail icons, or showing how complex scenarios are assembled together to form a more comprehensive 3D "puzzle".

C. Ubiquitous Language

Domain knowledge is important for capturing requirements [34]. Representing the domain with a ubiquitous language is an essential element for uncovering ambiguities and misunderstandings between software engineers and domain experts [35]. In our setting, it is essential to translate the language of the construction industry into the language of software engineers if a project is to succeed. Unified Modelling Language (UML), a software engineer's choice, is often inappropriate for communication with AECO professionals unfamiliar with that language [11], [17], [36].

[2]We originally entertained a thought to organize the levels in a *tree* instead of a *list of levels*, but the trees we explored were too confusing and cumbersome to use in practice, so we decided to use the simple hierarchy akin to the "height" of the viewer (ground-view *versus* bird-view).

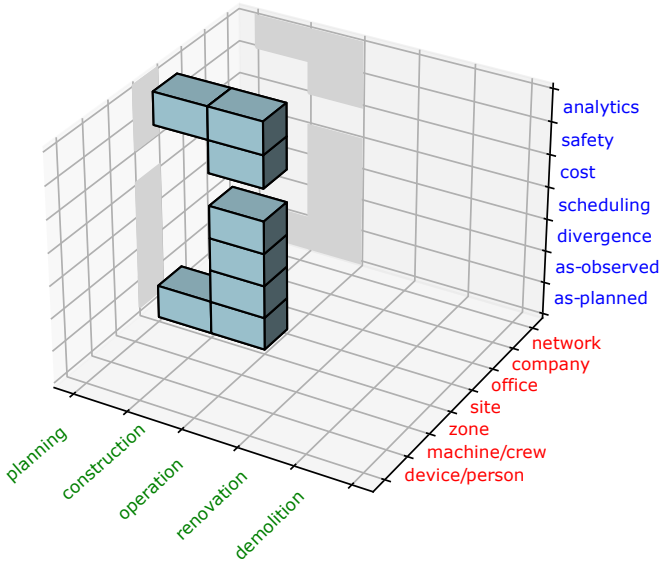

Fig. 3: Visualization of the scenario from Table Ib) "Risk Management" as a volumetric plot. Such a visualization is practical when scenarios are summarized as images. As we can see from the plot, the scenario "Risk Management" is focused on the site risks (such as injuries due to missing safety equipment and unfortunate site configurations or potential for fires due to spatio-temporal proximity of dangerous tasks). This scenario is not concerned about the individual site-independent risks related to device/person, machine/crew or zone, nor does it cover the risks that can affect the company or the network (*e.g.*, the risks concerning cost and scheduling). These risks are covered by a different scenario and were left out here for reasons of clarity.

Instead, we propose to use the **Building Information Model (BIM)** [5] as a formal language for capturing requirements and describing the system entities, relationships and behavior in our scenarios. Thankfully, the AECO industry enjoys the luxury of an increasing number of professionals getting versed in it world-wide (from Peru [37] to Malaysia [38]). Using BIM as a language for *software requirements* is novel as it is still predominantly regarded rather as an exchange or storage format, and to the best of our knowledge, it has not been used as ubiquitous language so far.

Though it is important to avoid ambiguities, it is equally important to avoid formalisms and additional notation whenever possible to lower the entry barrier and **"keep it simple"** [39]. Visual notation should be used sparingly and as comprehensible as possible [40]. In our experience (see Section VI), the AECO professionals lacking computer science background were unfamiliar with formalisms such as pseudo-code and structured text, and were rather put off and blocked to start the analysis too formally. Moreover, structuring the problem upfront can be detrimental as formalisms stifle creativity and impede the transfer of domain knowledge, while sense-making can nourish it [41]. We recommend starting with an abstract informal scenario and refining it only if utmost necessary.

Performance history is defined as an instance of `IfcPerfromanceHistory` and lives in the model `bim_extended`.

Cost is an instance of `IfcCostItem` living in the model `bim_extended`. It can be linked to *tasks* through GUIDs and `IfcRelAssignsToControl` (where the *task* is the related object).

Expenditure is an `IfcCostItem` living in the model `bim_extended` together with its relations.
To distinguish it from estimated *costs*, expenditures are explicitly linked to a performance history through `IfcRelAssignsToControl` (where the performance history is the control and the expenditure the related object).

Fig. 4: Example of a requirement from Table Ic) "Cost Tracking" where Building Information Modelling (BIM) helped us disambiguate the concepts and precisely express the relations in a way familiar to AECO professionals. Conventional approaches to requirements engineering use a general language such as UML which forces the invention of custom concepts and relations, and incurs the overhead since AECO experts not only need to learn a new language, but also need to understand these custom concepts and relations. In this example, costs and expenditures can be thought in many different ways, but framing them as `IfcCostItem`ś and their relations to other entities as `IfcRelAssignsToControl` make the reader immediately aware of intended hierarchies such as those stemming from `IfcControl`.

In our system scenarios [31], however, we did not find a single appropriate spot where requirements had to be formalized. What really worked well to clear up the ambiguity was reliance on BIM concepts such as identifying appropriate IFC classes and relationships. This helped us better discuss what data is relevant (and what can be left out), and often highlighted the data flow through our system. Figure 4 shows a requirement where BIM is used for more precise definitions.

In the future, we will develop a tool for the graphic formulation of scenarios to support less tech-inclined users with describing the scenarios using BIM, and composing simpler scenarios into more complex ones, to further minimize the need for formalisms.

D. Scenario Ontology

Scenarios span a spectrum from very simple action sequences (*e.g.*, sending warning messages) to extremely complex AECO processes (demolition and new construction of building structures, including rescheduling *etc.*). As their numbers increase, we need to organize and relate them to each other to obtain a holistic view of the requirements.

An ontology is a common strategy to organize requirements (*e.g.* [42], [23]).

To provide the reader with a big picture, we organize and present the relations between the scenarios in a *scenario ontology* following different vectors. We adopt the existing approaches: the scenarios can be related as steps of a sequence or a state-chart [42], or as abstract-concrete relations similar to sub-typing [33].

While the conventional approaches model the ontologies well and can be readily integrated for the AECO scenarios, our approach really shines when the ontology is defined in terms of the scenario space, where complex scenarios are **dissected by dimensions** of the scenario space. Instead of arbitrary criteria, we can systematically refine the ontology. We start from an initial set of scenarios and explore the space along its dimensions. We focus either on the individual aspects or phases in the building lifecycle of a scenario, or "zoom" in and out along the hierarchy levels of abstraction. This is akin to graphical exploration of [33], but in a coherent manner specifically tailored to the context of the AECO industry (see also Section VI-D).

Not all possible scenarios need to be defined in advance as the completeness of requirements is rarely feasible in practice [4]. Additional scenarios are added and embedded into the scenario space at a later point in time, as the discussion with the domain experts and other stakeholders progresses. This makes the scenario ontology a malleable structure, expected to evolve through the lifetime of the project in a systematic way. Depending on the adoption within a company and a wider community, certain parts of the ontology are shared across multiple projects. The most distilled parts are finally standardized to help requirements reuse [22]. Thanks to a standardized scenario space, it is possible to identify and situate such reusable parts, and consequently allows for consistent reading. Figure 5 gives an example of our refinement method and shows how a scenario is decomposed according to the phases of the building lifecycle.

E. Document Structure

While we do not consider the document structure to be crucial and believe that an alternative structure works equally well, we give a starting point for the practitioners here. Our main reference was [43], in particular their document analysis. The scenario document is generally structured by the aspects where each aspect is usually represented by a subsection. We note that not all aspects need to be represented and irrelevant aspects can be omitted when appropriate.

1) We start with a "Summary" section describing the scenario in succinct language.
2) The section "Relations to Other Scenarios" specifies how the scenario relates to other scenarios.
3) The section "Models" defines a mental map of the data, described in abstract terms. It sometimes includes relevant types, sources and the storage medium.
4) Third section, "Definitions", defines the entities as well as their relationships.

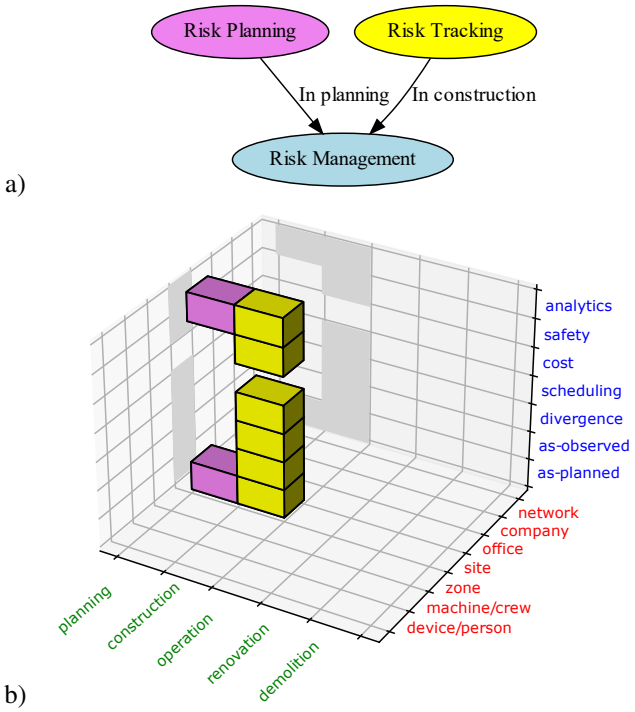

Fig. 5: **a)** The subset of the ontology showing how the scenario "Risk Management" from Table Ib) is decomposed into two scenarios, *Risk Planning* (violet) and *Risk Tracking* (yellow) along the phases "planning" and "construction". The arrows represent the relations. **b)** The decomposition of the scenario *Risk Planning* in the scenario space (see Section IV-B). The ontology gives the reader a first impression of the decomposition while the volumetric is an additional comprehension aid. The more focused scenarios, "Risk Planning" and "Risk Tracking", outlay the details of risk management concerning the respective phases. For example, the focus of "Risk Planning" is on inserting and defining risks, while "Risk Tracking" covers how the planned risks are observed and, if necessary, escalated. As we can immediately see in the volumetric, "Risk Planning" does not enforce safety measures, while "Risk Tracking" includes them (by the aspect "safety"). A standardized scenario space helps the writer to think in possible refinements and decompositions in a systematic way, and aids the comprehension of the reader at the same time.

5) The fourth section, "Scenario", elaborates the scenario with each aspect making a separate subsection. Not all aspects need to be represented and irrelevant aspects are omitted when appropriate.
6) The section "Test cases" describes how the implementation of the scenario is tested in practice.
7) The last section, "Acceptance Criteria", indicates the non-functional requirements such as efficiency constraints, expected magnitude of the data *etc.*

The document is appended by an index to the parts of the document marked with different phases in the building

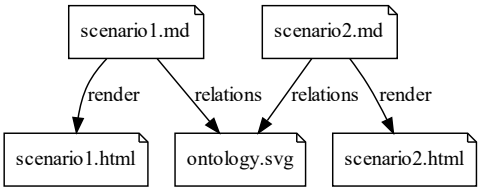

Fig. 6: Overview of the rendering process of our tool. The markdown files (`scenario1.md` and `scenario2.md`) are rendered to HTML (`scenario1.html` and `scenario2.html`). The scenario ontology (`ontology.svg`) is rendered as a graph based on the relations defined in the meta-data header in these two markdown files.

lifecycle and hierarchy levels of detail. We published the scenario documents in [31] and their source in [44].

V. Our Tool

Writing structured requirements is difficult and a specialized tool can provide some rails [45]. To that end, we implemented a prototype command-line application to help us analyze requirements in a scalable manner.

We chose markdown as the document format for its ease of editing in a text editor and its multi-media support [46]. The pseudo-code was marked with code blocks, while we used custom markups for markings (phases and hierarchy levels), references to definitions of models, entities, queries, commands and events. The references between the scenarios in the ontology were captured in a `<rasaeco-meta>` element with a property "nature" along with other meta information such as the volumetric in the scenario space. Since the source of the scenario document is in plain text, it can be readily fed into a version control system (such as Git [47]). We provide the source code of the scenario ontology tailored for our project [31] at [44].

The tool converts the individual source documents (in markdown) to browser-readable HTML documents. The set of all source documents is analyzed to reconstruct the ontology which is then also rendered as a graph (in HTML/SVG). Figure 6 outlines the processing of the data from the source to the artifacts.

The tool was implemented in Python 3.8 and released under MIT license on pypi.org. The standalone binaries are available for 64-bit Windows on the web site [8].

VI. An Experience Report on Applying RASAECO

In this paper we proposed a novel approach and a tool for analyzing and refining system scenarios in the AECO industry. Thus far, we focused our efforts on developing both the concepts and the tool to make them ready for practical use. Due to limited resources, we could not evaluate our approach on a set of different projects and recognize that more systematic field studies and varied use cases are needed to assess RASAECO in a more rigorous manner. Nevertheless, the initial results look promising. Inspired by the experience report from [28], we qualitatively assess how our approach helped us analyze the requirements for a large software project in AECO (see Section III):

- How did the proposed scenario space help with analyzing and structuring the scenarios?
- How did BIM as ubiquitous language affect the requirements analysis?
- How did our approach guide us with refining the scenarios (and how much effort could we re-use)?

A. The Requirements Analysis

Here we illustrate our requirements analysis so that other projects can be related to its volume, scope and characteristics.

The requirements elicitation phase took the first 6 months of the project (including brainstorming sessions *etc.*). We had trouble recognizing the requirements in the beginning. We started with a conventional approach, collecting the requirements and organizing them by functional/non-functional/system requirements. However, we faced multiple problems. The requirements were collected at different levels of abstraction, but we lacked a uniform notion of the levels and so had difficulties communicating within the team. The requirements were laborious to group and categorize due to the wide scope of the project, and we had a lot of initial dissonance on categories. Each contributor had their own notes, but the common grounding was difficult without a shared terminology. Finally, each contributor had her own scenario dimensions in mind which provoked further misunderstandings.

Faced with these challenges, we developed our approach to address them in a systematic way. The requirements analysis which we examine in this work was performed in the course of 3 weeks over 20 sessions at the end of this 6-months project phase, where each session took 2-2.5 hours. The sessions were guided by a software engineer (an author of the study, with more than 10 years experience in software engineering, but no prior experience in AECO or related fields), where he interviewed 4-5 stakeholders relevant for the respective system scenario. Each session covered a specific topic (*e.g.*, "Truck Guidance" summarized in Table Ia), and what took place was something between an interview and a conversation. As the sessions progressed, the previous scenarios were refined and refactored in addition to creating new ones. The final scenarios comprised the whole scope of our system.

The analysis finally resulted in 20 system scenarios. Most scenarios were rather short (8 scenarios with < 500 words). There were medium-long scenarios (7 with up to 1000 words) and a few long ones (5 with more than 1000 words). Coincidentally, the long scenarios (> 1000 words) encompassed the core modules of the system (such as "Digital Reconstruction", "UXV Recording" and "Virtual Inspection") with fewer AECO-specific elements. In contrast, AECO-specific scenarios, such as "Risk Management", could be further refined by our approach into smaller sub-scenarios, additionally resulting in a smaller word count (all less than 1000 words).

The final ontology graph was not highly connected and remained manageable. With respect to incoming edges, many scenarios had no incoming edge (9 scenarios). There were a few scenarios with one or two incoming edges (5 and 4, respectively). There was only one scenario with 3 and one with 4 incoming edges, respectively. The outgoing edges were very few per scenario: some with zero edges (3 scenarios), most had a single outgoing edge (15), and only two scenarios had two and three outgoing edges, respectively. Notably, the scenarios with many incoming edges included a core module ("Topic Management", serving as a communication bus) and a complex scenario refined in multiple sub-scenarios ("Risk Management"). The out-degree of the scenarios with more than one outgoing edge ("Digital Reconstruction" and "Risk Tracking") reflects their data flows (pushing the information to multiple downstream scenarios).

Constricted by the scope of the project, the volume of the scenarios is concentrated in the planning and construction phases (only 1 about planning, 14 about construction and 5 about both phases). The abstraction levels reached "office" and spanned all aspects. A sub-graph of our ontology in Figure 7 gives a glimpse of the scope and complexity of our ontology to be compared with other projects. We discuss the generalizability to other AECO projects in Section VI-E.

B. Effects of the Scenario Space on Analysis

Here we observe the effects that the scenario space had on writing the individual scenarios.

Degree of participation. Our approach forced a certain structure on the authors, so that they had to at least be familiar with the relevant terms such as dimensions of the scenario space such as aspects, phases and levels. Somewhat unexpectedly, thinking and writing in such structures proved to be too much of a barrier for AECO experts, so a software professional had to take over that task. The resulting text was readable and something they could comment on and finally approve of, but could not contribute to freely.

Discovering AECO-specific complexities. The relatively rigid structure forced on us an early positive confrontation between what is easy/difficult to implement and what is of low/high value from an AECO perspective as the structure prevented us to ignore the different dimensions of the solution space. We experienced that our approach substantially framed our thinking in the terms of the solution space. In particular, the tension between as-planned, as-observed and their divergence was very useful as it mapped well to many of the cyber-physical problems (key to 7 out of 20 scenarios) and forced us to define the sources of the data at an early stage. Notably, the risk-related scenarios ("Risk Tracking", "Fire Risk" and "Fall Risk") and delivery scenarios ("Truck Guidance" and "Crane Guidance") could easily be mapped to this planned-observed schema.

Additionally, the distinction between cost, scheduling, analytics and safety helped us to distinguish the nuances and to group the requirements. For example, while working on the scenarios related to logistics, we struggled with the opaque

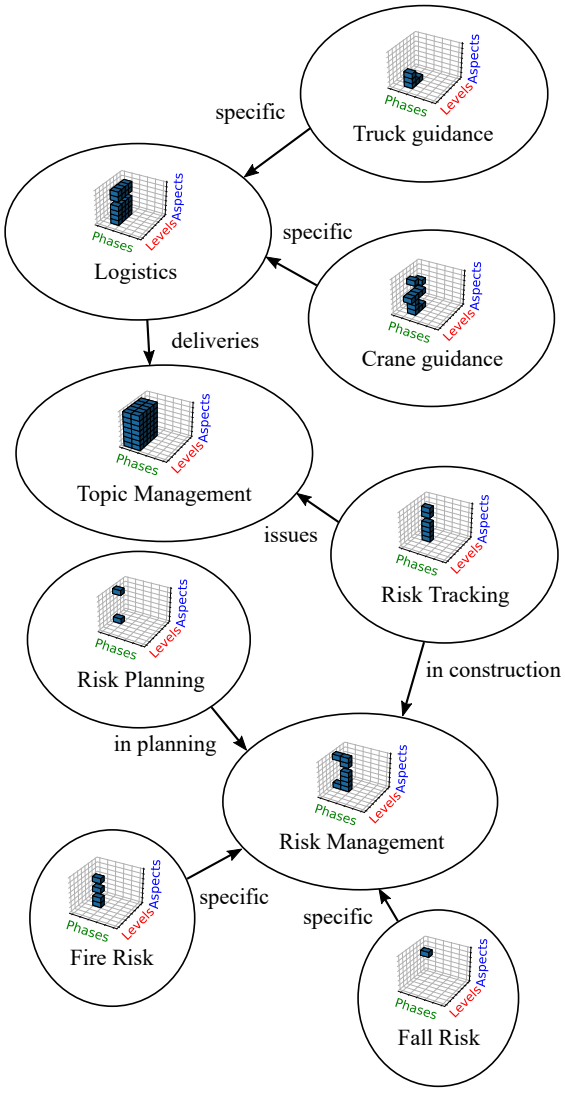

Fig. 7: The sub-graph of our ontology. The full ontology is available at [31]. This sub-graph gives a glimpse of the scope and complexity of our scenarios to be compared with similar ontologies. The volumetric plots are iconified intentionally and are not meant to be read in detail but rather to give a vague idea about the concerns of a scenario and its focus. A larger volumetric plot, such as one given in Figure 3 is included in the text of the scenario.

two-fold nature of the concept "missed deadlines". Distinguishing explicitly between scheduling and analytics finally identified the "missed deadlines" as an analytics indicator and a separate trigger for re-scheduling. In a similar vein, the phases allowed us to think about the temporal sequence of the logistics workflow: entrances and exits are defined during planning (and rarely updated later), while delivery locations change frequently during the construction phase.

For us this indicates that our method is well-suited to drilling down into important details through deep discussions at the core of the structured scenarios. As the structure forced

us to quickly identify and clarify important *AECO* unknowns, unclear parts of the AECO procedure were revealed. However, the experiences of using the method in additional projects is still needed to obtain better validation.

Markings as a disambiguation instrument. We used semantic markings sparingly, as additional colors and superscripts generally clutter the text (see Figure 1 and 2), although occasionally they were useful tools to explicitly disambiguate parts of a sentence or paragraphs.

We used level markings more often than phase markings. Most scenarios did not need level markings (11 scenarios), some needed 1-6 level markings (8 scenarios), and one scenario ("UXV Recording") needed 12 level markings. In that particular scenario the level markings helped us disentangle a "ball of yarn" spun between devices and operating stations.

Phase markings were used scantly. On the one hand, most scenarios dispensed of phase markings completely (16 scenarios), and very few used them sparingly (3 scenarios contained one or two phase markings). On the other hand, one scenario ("Evolving Plan") specifically benefited from them and used as much as 5 markings to highlight the phases in the text, owning to the scenario scope (plan evolution through different phases).

We thus observe that the markings are an important tool in the toolbox, but one which is to be used selectively.

C. Advantages of BIM as Ubiquitous Language

We made a novel choice to use BIM (and its open standard, IFC) as ubiquitous language (see Section IV-C). This meant that we could often use terms which were well defined from a software engineering perspective, but also well known by many of the AECO professionals as BIM is used widely in construction projects, and a lot of today's AECO software tools and collaboration practices support it.

For example, it might be fuzzy to talk about a "named 3D area" on the construction site. Instead, we used `IfcZone` which was clearly understood. `IfcZone` is exactly what one can require to be included in the IFC files that will be used as some of the inputs in the software tools that we will develop.

Furthermore, we chose to use IFC terms for concepts which are not commonly stored in IFC files because we wanted to take advantage of the clear definition existing in the IFC schema. One example for this is the entity `IfcTask`. When we referred to this entity, we knew that it was a task, that it had a start and an end time, that it was part of a workflow and that it could be related to a specific set of building elements. Yet another example is `IfcActor` entertaining well-defined and documented relations such as belonging to an organization or being responsible for a task.

In our 20 scenarios, we specified 98 definitions. We explicitly did not strive to formalize all of them, and did so only when necessary to avoid ambiguities (see Section IV-C). In total, this gave us only 64 formal definitions (65%)[3]. Out of these 64 formal definitions, we matched 36 (56%) to an IFC entity, while the remainder (28, 44%) were custom data entities specific to our project[4]. In the light of these numbers, we infer that BIM is not a silver bullet, but it provided us with higher precision and easier communication in a substantial fraction of definitions.

D. Effects of the Scenario Space on Scenario Refinement

During the requirements analysis, we encountered many cases where the requirements were reshuffled, regrouped or abstracted and re-specified, similar to how a programmer refactors the source code. On one side, some of the refactorings such as concretization of the scenario "Risk Management" into "Fall Risk" and "Fire Risk" (see Figure 7), were straightforward to derive and we did not leverage our approach to systematic refinement. In two notable cases, however, systematic exploration of the scenario space helped us refine the existing requirements in an efficient way so we describe them here as cases in point.

The first case is the dissection of the scenario "Risk Management" by the dimension "phase". As the scenario grew larger and unwieldy, we needed to refactored it into smaller parts both for readability and for the ease of future re-writings. While multiple split axes came into question, using the phase felt the most natural. Accordingly, the subsequent dissection was fast. Though it is difficult to compare the refactoring to writing a scenario from scratch, we remark that it took us only 30 minutes to split it up into two ("Fall Risk" and "Fire Risk") and write an overview in a bundling scenario (a new "Risk Management"). The split-up can be observed in Figure 5.

The second case concerns the introduction of a novel scenario, "Crane Guidance" after already writing a scenario on a similar subject, "Truck Guidance" (see Table Ia). The markings in the text helped us move the general bits (marked with the level "site" and "office") to an abstract scenario "Logistics", while we kept the truck-specific details (at the level "machine/crew") in "Truck Guidance". Afterwards, we added the novel scenario "Crane Guidance", where we filled in the specifics for the crane deliveries. Analogous to the first refinement, we again measured a shorter session until satisfactory results (only 1 hour compared to 2-2.5 hours for a completely novel scenario). Figure 8 illustrates the refinement.

While more cases are necessary for a solid evaluation, the two cases presented here suggest that our refinement strategy is beneficial for efficiency and fosters re-use both of concrete requirements and mental efforts. In our subjective opinion, we find that our approach strikes a good balance between the rigidness of the structure, compactness, readability, re-usability and modifiability.

E. Generalizability to Other Projects

Our approach was developed based on the experiences during a single project. Further studies are necessary to establish its generalizability to other projects. However, we observe that

[3]The term "formal" is used here in a relaxed way. A formal definition specified properties of an entity, where property types were included in non-obvious cases.

[4]We also counted a definition as matching an IFC entity if it specified a super-definition already defined using an IFC entity as in those cases repeating the IFC entity would have been redundant.

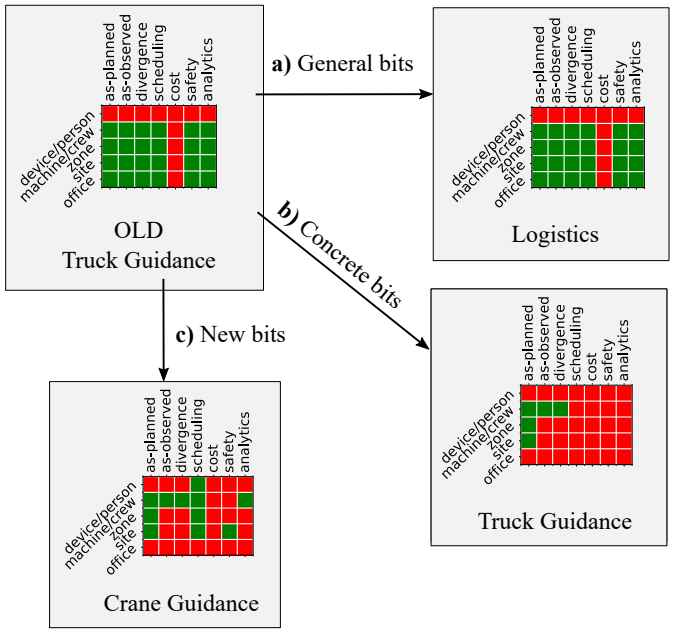

Fig. 8: Introduction of a novel scenario, "Crane Guidance" through refactoring of the scenario "Truck Guidance" from Table Ia). As the scenarios cover only the phase "construction", we omit the corresponding dimension for readability. **a)** We extracted the general bits from levels "site" and "office" from the original scenario "Truck Guidance" to the scenario "Logistics". **b)** We refactored the details at the level "machine/crew" to the new scenario "Truck Guidance". **c)** We wrote crane-specific requirements in the new scenario "Crane Guidance". The markings in the original scenario and the mental framework of the scenario space reduced the effort for introduction of "Crane Guidance" to only 1 hour, compared to 2-2.5 hours usually needed for novel scenarios.

the conventional approaches (such as [6], [11], see Section II and VI-A) provided little guidance for the complex nature of the project which surpasses the data governance and ventures into cyber-physical systems. We anticipate that other AECO projects entailing system interactions with the physical world face similar challenges, which in parallel is also observed in the field of smart manufacturing creating similar solutions [3].

We do not claim the fallibility of the conventional approaches by any means. They are documented to work well for AECO systems concerning data governance, and it is indeed questionable whether such systems need tools such as a scenario space. On the other hand, for the development of cyber-physical systems like ours (see Section VI-A), we believe that we contribute to the literature by documenting our method, our experience using it and its viability.

VII. Threats to Validity

Our investigation carries some threats to validity as formulated by [48] regarding the interpretation of the results.

The method proposed here describes how one can capture structured system scenarios in a way that is well suited for complex multi-disciplinary and heterogeneous cases in the AECO industry. In the research project [27] where this method was used, we experienced this as an efficient and well-suited way of capturing system requirements. But should this method be used by others? It is very important to consider to what degree our positive experiences of using this method can be used as a evidence for a good fit for other projects.

Regarding **internal validity** our approach was created as a result of experiences of working with software design and capturing AECO industry needs and improvement potentials over many years. However, it is important to notice that this method has not been tried out in external projects. For that reason, our paper should be seen as a case study of our experiences. Further studies would be needed to know with more certainty if this method works well in other projects.

When it comes to **external validity**, there was especially one unusual condition of the project. We worked with experts from many domains on workflows which were rather innovative for their line of business. If a project is less innovative, more data-oriented and with little or no interactions with the physical world, the benefits of our approach are questionable as conventional approaches (see Section II) are reported to work well. Our approach is likely to generalize to projects with similar characteristics (*e.g.*, cyber-physical systems of similar scope as described in Section VI-A). Whether it generalizes beyond these limits remains an open question.

VIII. Conclusion and Future Work

We presented an approach and an accompanying tool for requirements analysis specific to the context of the AECO industry. The requirements are captured based on scenarios living in a scenario space (along the three dimensions: aspects, phases in a building life cycle and hierarchy levels of details), while the relations between the scenarios are modeled as an ontology. We developed a command-line program to aid requirements analysis and facilitate the (re-)rendering of the scenario documents. Finally, we evaluated our approach on 20 practical AECO scenarios from a large AECO project.

In future work, we would like to explore how patterns of common AECO scenarios and relations emerge and how cataloguing them facilitates requirements reuse and accelerates the requirements analysis. In additional experiments, we would like to validate more thoroughly the adequacy of the scenario space, both its dimensions and its axes. Finally, a study on a larger scale needs to be performed to assess what individual factors of the approach lead to improvement and degradation of the requirements analysis, respectively, and how our approach quantitatively compares to other methods.

Acknowledgement

This work is part of the BIMprove Project, an initiative funded by the European Union's Horizon 2020 Research and Innovation programme under grant agreement N° 958450, with Focus Area Building a low-carbon, climate resilient future.

REFERENCES

[1] W. Shen, Q. Hao, H. Mak, J. Neelamkavil, H. Xie, J. Dickinson, R. Thomas, A. Pardasani, and H. Xue. Systems integration and collaboration in architecture, engineering, construction, and facilities management: A review. *Advanced Engineering Informatics*, 24, 2010.

[2] J. Dick, E. Hull, and K. Jackson. *Requirements Engineering*. Springer International Publishing, 2017.

[3] R. Heidel, M. Hoffmeister, M. Hankel, and U. Döbrich. *The Reference Architecture Model RAMI 4.0 and the Industrie 4.0 component*. VDE Verlag, 2019.

[4] Martin Glinz. Improving the quality of requirements with scenarios. In *Second World Congress on Software Quality*, 2000.

[5] Industry foundation classes (IFC) for data sharing in the construction and facility management industries — part 1: Data schema. Standard ISO 16739-1:2018, International Organization for Standardization, 2018.

[6] Eissa Alreshidi, Monjur Mourshed, and Yacine Rezgui. Cloud-based BIM governance platform requirements and specifications: Software engineering approach using BPMN and UML. *Journal of Computing in Civil Engineering*, 30, 10 2015.

[7] H. Gaspard-Boulinc and S. Conversy. Usability insights for requirements engineering tools: A user study with practitioners in aeronautics. In *IEEE International Requirements Engineering Conference (RE)*, 2017.

[8] RASAECO tool. Retrievable from https://github.com/mristin/rasaeco. Accessed: 2021-04-01.

[9] F. Cousins. Starchitects and jack-hammers: Requirements engineering challenges and practices in the construction industry (keynote). In *IEEE International Requirements Engineering Conference (RE)*, 2013.

[10] buildingSMART use case management. Retrievable from https://ucm.buildingsmart.org/. Accessed: 2021-04-01.

[11] Yusuf Arayici, Vian Ahmed, and Ghassan Aouad. A requirements engineering framework for integrated systems development for the construction industry. *Electronic Journal of Information Technology in Construction (ITCon)*, 11, 03 2006.

[12] E. Alreshidi, M. Mourshed, and Y. Rezgui. Requirements for cloud-based BIM governance solutions to facilitate team collaboration in construction projects. *Requirements Engineering Journal (REJ)*, 23, 2018.

[13] Muhammad Shafiq, Jane Matthews, and Stephen Lockley. A study of BIM collaboration requirements and available features in existing model collaboration systems. *Electronic Journal of Information Technology in Construction (ITcon)*, 18, 08 2013.

[14] C.-E. Tolmer, C. Castaing, Y. Diab, and D. Morand. Adapting LOD definition to meet BIM uses requirements and data modeling for linear infrastructures projects: using system and requirement engineering. *Visualization in Engineering*, 5, 12 2017.

[15] S.-C. Chien and A. Mahdavi. Requirement specification and prototyping for user interfaces of buildings' environmental controls. *Electronic Journal of Information Technology in Construction (ITcon)*, 14, 2009.

[16] P. B. Purup and S. Petersen. Requirement analysis for building performance simulation tools conformed to fit design practice. *Automation in Construction*, 116, 2020.

[17] O. Berard and J. Karlshoej. Information delivery manuals to integrate building product information into desgin. *Electronic Journal of Information Technology in Construction (ITcon)*, 17, 2012.

[18] A. Akanmu, C. Anumba, and J. Messner. Scenarios for cyber-physical systems integration in construction. *Electronic Journal of Information Technology in Construction (ITCon)*, 18, 2013.

[19] F. Correa. Cyber-physical systems for construction industry. In *IEEE Industrial Cyber-Physical Systems (ICPS)*, 2018.

[20] W. Meincke. Requirements in the loop - a computer-aided analysis of consistency, completeness, and correctness of requirements. In *IEEE International Requirements Engineering Conference (RE)*, 2020.

[21] W. Alhoshan, R. Batista-Navarro, and L. Zhao. Towards a corpus of requirements documents enriched with semantic frame annotations. In *IEEE International Requirements Engineering Conference (RE)*, 2018.

[22] X. Franch, C. Palomares, and C Quer. Industrial practices on requirements reuse: An interview-based study. In *Requirements Engineering: Foundation for Software Quality (REFSQ)*, 2020.

[23] H. Nguyen, J. Grundy, and M. Almosy. Guitar: An ontology-based automated requirements analysis tool. In *IEEE International Requirements Engineering Conference (RE)*, 2014.

[24] M. Fockel and J. Holtmann. ReqPat: Efficient documentation of high-quality requirements using controlled natural language. In *IEEE International Requirements Engineering Conference (RE)*, 2015.

[25] Zahra Shakeri, Mohammad Noaeen, and Guenther Ruhe. Requirements engineering visualization: A systematic literature review. In *IEEE International Requirements Engineering Conference (RE)*, 2016.

[26] Y. D. Pham, A. Bouraffa, and W. Maalej. Shapere: Towards a multi-dimensional representation for requirements of sustainable software. In *IEEE International Requirements Engineering Conference (RE)*, 2020.

[27] BIMprove Project. Retrievable from https://www.bimprove-h2020.eu. Accessed: 2021-04-01.

[28] J. Cleland-Huang and M. Vierhauser. Discovering, analyzing, and managing safety stories in agile projects. In *IEEE International Requirements Engineering Conference (RE)*, 2018.

[29] M. Bano, D. Zowghi, A. Ferrari, P. Sploetini, and B. Donati. Learning from mistakes: An empirical study of elicitation interviews performed by novices. In *IEEE International Requirements Engineering Conference (RE)*, 2018.

[30] A. Koutamanis. Dimensionality in BIM: Why BIM cannot have more than four dimensions? *Automation in Construction*, 114, 2020.

[31] BIMprove Scenario Ontology. Retrievable from https://mristin.github.com/bimprove-scenarios. Accessed: 2021-04-01.

[32] RIBA plan of work. Retrievable from https://www.architecture.com/knowledge-and-resources/resources-landing-page/riba-plan-of-work. Accessed: 2021-04-01.

[33] P. Ghazi and M. Glinz. An experimental comparison of two navigation techniques for requirements modeling tools. In *IEEE International Requirements Engineering Conference (RE)*, 2018.

[34] A. M. Aranda, O. Dieste, and N. Juristo. Effect of domain knowledge on elicitation effectiveness: An internally replicated controlled experiment. *IEEE Transactions on Software Engineering*, 42, 2016.

[35] E. J. Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison Wesley Longman, 2003.

[36] M. Glinz. Problems and deficiencies of UML as a requirements specification language. In *International Workshop on Software Specification and Design*, 2000.

[37] C. Sanchís-Pedregosa, J. Vizcarra Aparicio, and A. Leal-Rodríguez. BIM: a technology acceptance model in Peru. *Electronic Journal of Information Technology in Construction (ITCon)*, 25, 2020.

[38] W. Enegbuma, G. Aliagha, and K. Ali. Preliminary building information modelling adoption model in Malaysia. *Construction Innovation*, 14, 2014.

[39] Alistair Mavin, Philip Wilkinson, Sarah Gregory, and Eero Uusitalo. Listens learned (8 lessons learned applying EARS). In *IEEE International Requirements Engineering Conference (RE)*, 2016.

[40] P. Caire, N. Genon, P. Heymans, and D. L. Moody. Visual notation design 2.0: Towards user comprehensible requirements engineering notations. In *IEEE International Requirements Engineering Conference (RE)*, 2013.

[41] P. Ralph and R. Mohanani. Is requirements engineering inherently counterproductive? In *IEEE/ACM International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks)*, 2015.

[42] M. Glinz. An integrated formal model of scenarios based on statecharts. In *European Software Engineering Conference (ESEC)*, 1995.

[43] D. Rapp, A. Hess, N. Seyff, P. Spörri, E. Fuchs, and M. Glinz. Lightweight requirements engineering assessments in software projects. In *IEEE International Requirements Engineering Conference (RE)*, 2014.

[44] BIMprove Scenario Ontology (source code). Retrievable from https://github.com/mristin/bimprove-scenarios. Accessed: 2021-04-01.

[45] G. Lucassen, F. Dalpiaz, J. v. d. Werf, and S. Brinkkemper. Improving agile requirements: the quality user story framework and tool. In *IEEE International Requirements Engineering Conference (RE)*, 2016.

[46] S. Leonard. Guidance on markdown: Design philosophies, stability strategies, and select registrations. Technical report, Internet Engineering Task Force (IETF), 2016.

[47] S. Chacon and B. Straub. *Pro Git*. Apress, 2014.

[48] Robert K. Yin. *Case study research and applications: Design and methods*. Sage publications, 2017.