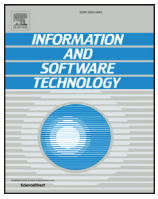
Information and Software Technology 121 (2020) 106275

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



A survey on the practical use of UML for different software architecture viewpoints

Mert Ozkaya[a,*], Ferhat Erata[b]

[a] Yeditepe University, Istanbul, Turkey
[b] Yale University, New Haven, CT, USA

ARTICLE INFO

Keywords:
Software architecture viewpoints
UML
Survey
Practitioners

ABSTRACT

Context: Software architecture viewpoints modularize the software architectures in terms of different viewpoints that each address a different concern. Unified Modeling Language (UML) is so popular among practitioners for modeling software architectures from different viewpoints.
Objective: In this paper, we aimed at understanding the practitioners' UML usage for the modeling of software architectures from different viewpoints.
Method: To this end, 109 practitioners with diverse profiles have been surveyed to understand practitioners' UML usage for six different viewpoints: functional, information, concurrency, development, deployment, and operational. Each viewpoint has been considered in terms of a set of software models that can be created in that viewpoint.
Results: The survey includes 35 questions for different viewpoint models, and the results lead to interesting findings. While the top popular viewpoints for the UML-based software architecture modeling are the functional (96%) and information (99%) viewpoints, the least popular one is the operational viewpoint that is considered by 26% of the practitioners. The top popular UML modeling tool is Enterprise Architect regardless of the viewpoints considered. Concerning the software models that can be created in each viewpoint, UML's class diagram is practitioners' top choice for the functional structure (71%), data structure (85%), concurrency structure (75%), software code structure (34%), and system installation (39%), and system support (16%) models; UML's sequence diagram is the top choice for the data lifecycle models (47%); UML's deployment diagram for the physical structure (71%), mapping between the functional and physical components (53%), and system migration (21%) models; UML's activity diagram for the data flow (65%), software build and release processes (20–22%), and system administration (36%) models; UML's component diagram for the mapping between the functional and concurrent components (35%), software module structure (47%), and system configuration (21%) models; and UML's package diagram for the software module structure (47%) models.

1. Introduction

Software systems that are developed are getting larger and more complex each day due to the ever-increasing customer demand and the improving technology. This also leads to complex software system designs that cannot easily be managed by the project stakeholders and cause the software systems to be delivered late (or over-budget) or developed wrongly. To manage the software design complexity, the notion of software architecture has been proposed in the early nineties, which promotes the decompositions of large and complex software systems into manageable components and connectors and their high-level reasoning [1–3]. Software architectures can be specified in a more modular and thus understandable way in terms of different perspectives that each concern different aspects of software development. These perspectives are actually called as software architecture viewpoints, which have been initially proposed in the mid-nineties by Kruchten [4]. Kruchten separates the software architecture design into four viewpoints, which are the logical, process, development, and physical viewpoints. In the logical viewpoint, the software system to be built is decomposed into components and their relationships. In the process viewpoint, the concurrent interactions of the system components are focused upon. In the development viewpoint, the implementation-related issues are focused upon. Lastly, in the physical viewpoint, the physical hardware units and their physical connections are focused upon, in which the logical components are to be allocated. Another important work has been performed by Soni et al. [5], who proposed the conceptual, module, execution, and code viewpoints. The conceptual viewpoint is essentially the same as Kruchten's logical viewpoint, and the module viewpoint adapts the conceptual viewpoint with the high-level implementation design decisions that are independent of any programming languages (e.g., the

* Corresponding author.
E-mail address: mozkaya@cse.yeditepe.edu.tr (M. Ozkaya).

https://doi.org/10.1016/j.infsof.2020.106275
Received 16 November 2018; Received in revised form 12 January 2020; Accepted 3 February 2020
Available online 4 February 2020

ideal structure of the software implementation). The execution viewpoint is for modeling the run-time elements (e.g., operating system and process) and their communications. The code viewpoint represents the organisation of the source code into modules in any programming languages. Later on, Clements et al. [6], offered the component & connector, module, and allocation viewpoints that are each modeled in terms of a set of architectural styles. The component & connector viewpoint focuses on decomposing software systems into components that interact via connectors, while the module viewpoint focuses on the software implementation and the allocation viewpoint focuses on the physical system units. Garland et al. [7] also offered a large set of viewpoints that are categorised into five groups: domain analysis, component design, sub-system design, data design, process & deployment design. The domain analysis focuses on the top-level architecture description of systems. The component design focuses on the components composing the systems, their interactions, and behaviors. The sub-system design focuses on systems's build-time and organisational structures. The data design focuses on data architecture and the process & deployment design focuses on the concurrency issues and the physical system structures. The IEEE community standardized the concept of software architecture viewpoints [8] (i.e., IEEE Standard 1471). They treat the view and viewpoint concepts separately.[1] That is, each view is essentially a set of software models that describe the software system from a particular viewpoint. The view models are expected to satisfy the rules and constraints of the viewpoint that define the view.

Software architectures can be modeled in terms of different viewpoints using the software modeling languages, which can be architecture description languages [9], domain-specific languages (e.g., Liszt [10], HIPA [11], and NDL [12]), and UML [13]. Among them, Unified Modeling Language is considered as the top-used language by practitioners for modeling software architectures [14–17]. UML offers various diagrams for the visual specifications of software systems, which can be categorised into static and dynamic diagrams. The static diagrams are concerned with the system structures while the dynamic diagrams are concerned with the system's changing behaviors. UML's static diagrams are the composite structure diagram, deployment diagram, package diagram, profile diagram, class diagram, object diagram, and component diagram. UML's dynamic diagrams are the activity diagram, use case diagram, state machine diagram and some interaction diagrams that are the sequence, communication, and timing diagrams. UML is supported by a huge number of modeling tools, e.g., Enterprise Architect, Visual Paradigm, MagicDraw, Modelio, IBM Rational Rhapsody, etc. Using these tools, practitioners can model their software systems in UML and perform many different operations such as model management, analysis, simulation, user collaboration, project management, model transformation, documentation, etc. UML has also been extended many times for adapting UML to different domains (e.g., embedded systems, multi-agent systems, distributed systems, and web applications) [18].

1.1. Motivation and goal

Given UML's huge support by the community (including the tool vendors and language developers) and its popularity among practitioners, one would expect to learn to what extent practitioners use UML and its various types of diagrams for the modeling of software architectures from different viewpoints. Unfortunately, as discussed in Section 2, the existing empirical studies on UML do not aid in understanding practitioners perspectives on using UML for different architecture viewpoints. While it is possible with the existing literature to get informed about many practical issues on UML such as practitioners' motivation/demotivation on UML and its diagram types, UML's evaluation for some quality properties, and UML's usage rate for particular aspects of software development, one may not easily understand to what extent practitioners use UML for various architecture viewpoints that each address a different set of concerns of software development. Indeed, it is not clear as of now which viewpoints are modeled with UML in practice, the specific concerns addressed with UML in each viewpoint, practitioners' choice of the UML diagrams for each concern, and practitioners' choices of the UML modeling tools.

To form the basis of our research precisely, we initially define the meta-model in Fig. 1 with the inspiration of the IEEE standard 1471, which shows the abstract concepts on describing software architectures from multiple viewpoints and the relationships between those concepts. The software architecture descriptions are specified with the software architecture description languages (e.g., UML), which offer a modeling notation set and are supported with some modeling editors for the practitioners to specify the architecture descriptions via the notation set. An architecture description is essentially specified in terms of a set of views that each derive from a viewpoint which deals with a relevant (i.e., cohesive) set of concerns. The viewpoints herein are proposed by the viewpoint frameworks that modularise the architectural descriptions in terms of different viewpoints by considering the needs of particular types of software systems or domains. A view of any viewpoint is represented with a set of model types that each describe how to solve a particular concern of the relevant viewpoint. The model types herein are instantiated as the view models, which are specified via the modeling notation sets that are offered by the architecture description languages. As discussed in the next paragraph, we use our meta-model definition to determine the concrete techniques, approaches, and tools in our own context that match with and satisfy the abstract concepts of the meta-model definition and show how we consider in this research the architectural descriptions from multiple-viewpoints. By doing so, we essentially aimed to establish the goal of our research in this paper in a precise way that can clearly be understood in terms of its components.

Given the meta-model of our research in Fig. 1, we consider UML for the software architecture description language. Also, to determine the set of viewpoints, we consider Rozanski et al.'s approach [19] for the architecture viewpoint framework, which focuses on the needs of practitioners working with any types of information systems and at the same time supports the concepts of viewpoint, view, and model type that match well with our meta-model definition. So, our goal in this research is to survey the practitioners to understand to what extent practitioners use UML for describing software architectures from different viewpoints that Rozanski et al. offers in their framework. We used Rozanski et al.'s six different architecture viewpoints in this study, which are the functional, information, concurrency, development, deployment, and operational viewpoints. The functional viewpoint is concerned with the functional elements that compose the software systems and their interactions. The information viewpoint is concerned with how the system data are defined, stored, accessed, and transmitted. The concurrency viewpoint is concerned with mapping the functional elements into the concurrent elements and their concurrent interactions. The development viewpoint is concerned with modeling the plans and decisions made about the software development process such as structuring the software code and planning the software build & release processes. The deployment viewpoint is concerned with the physical structure of systems that represent the hardware elements in which the functional elements will be run and their physical relationships. The operational viewpoint is concerned with the operational issues that occur while the system is run on its production environment and deal with monitoring, administering, restoring, and supporting the system. Note here that the operational viewpoint focuses on managing and controling the running

[1] The view and viewpoint terms are each referred explicitly in the rest of the paper.

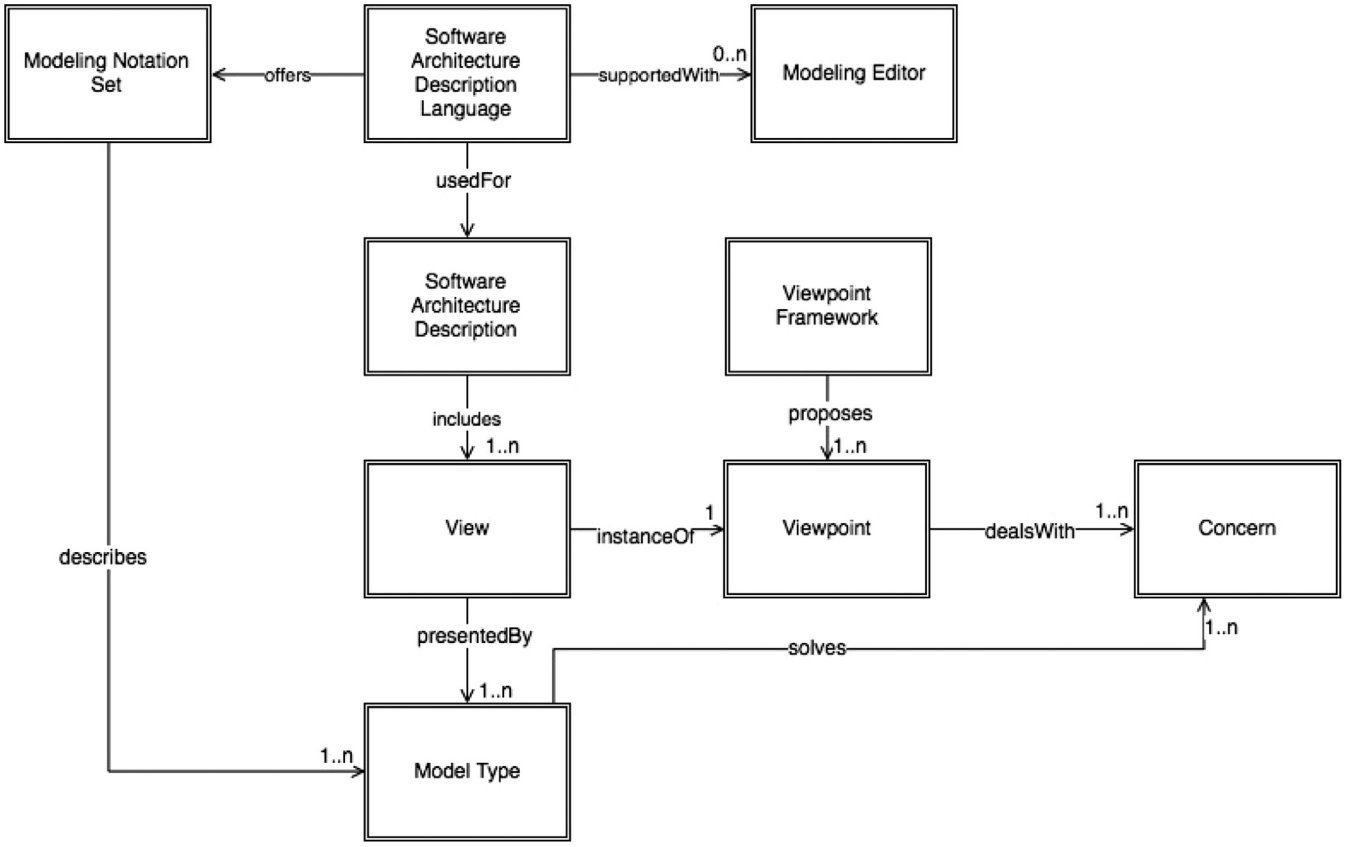


Fig. 1. The meta-model definition of describing software architectures from multiple viewpoints.

system rather than the system's requirements and design activities that other viewpoints focus on. However, the operational issues are highly crucial and need to be resolved early on so as to minimise any required effort that may get much bigger if the operational issues were dealt with while the system is run in its environment. In Table 1, we show the relationships between Rozanski et al.'s viewpoints, the model types that we consider from Rozanski et al.'s model list for each viewpoint, the UML diagrams to be used as the modeling notations for instantiating the model types, and the UML CASE tools to be used as the model editors for the UML language. Note here that Rozanski et al.'s viewpoint model types that we consider are each described in Section 5, where we discuss the survey analysis results for each viewpoint and its model types. The set of UML diagrams and UML tools herein have been determined via the feedback received from the survey's pilot study, which have been conducted among a set of practitioners before releasing the survey in the way discussed in Section 4.1.

With our survey, we essentially intend to shed light on many important but neglected issues including which viewpoints are important for practitioners in their UML modeling, how that is affected by practitioners' work industries, what types of models practitioners tend to specify with UML for representing their views in each viewpoint, the UML diagrams that practitioners frequently use in their model specifications in each viewpoint, and practitioners' preferences for the UML CASE tools. The survey results will be essentially useful for attracting the interest of the software engineering community towards the multiple-viewpoints modeling for managing large and complex software systems and its practical use via the de-facto UML language. Those who wish to engineer their own domain-specific languages will be able to use the survey results to determine the popular viewpoints in their domain of interest and the popular UML notations used for specifying different types of models that conform to those viewpoints. So, the language engineers can easily address in their languages the expectations of practitioners in terms of the multiple-viewpoints modeling and the modeling notations to be used. Moreover, the results will certainly help the tool vendors to improve their UML tools or build new tools in a way that addresses the issues about the multiple-viewpoints modeling determined via the survey. The survey results may further trigger new researches such as the analysis of the existing UML tools or the DSL meta-modeling tools for the multiple-viewpoints support. So, practitioners in different industries may use those analysis results to determine many important requirements, including (*i*) the tools' support for any of the viewpoint approaches discussed above (e.g., Kruchten's model), (*ii*) the tools that offer their own viewpoints, or (*i*) the tools that allow practitioners to define user-specific viewpoints.

1.2. Paper structure

In the rest of this paper, firstly, we discuss the related works with our study. Then, the research questions that our survey aims to answer are given, which is followed by the discussions of the research methodology of our survey that includes the survey design, execution, and sampling. After that, the survey results are discussed for each software architecture viewpoint considered. Lastly, the summary of findings, the lessons learned, and any threats to the validity of the results are discussed.

2. Related work

In this section, the studies that are relevant to our survey study are discussed. We firstly focus on the software architecture viewpoint frameworks that are similar to Rozanski et al.'s framework which are survey is based on. Following that, we focus on the analytical studies on the UML language.

Table 1
Describing the relationships between Rozanski et al.'s viewpoints and model types, UML's diagrams, and the UML tools.

Viewpoint	Model types	UML diagrams	UML tools
Functional	Functional structure	Class, Component, Composite Structure, Deployment, Object, Package, Profile	ArgoUML, BoUML, Enterprise Architect, MagicDraw, MS Visual Studio, Obeo UML Designer, Modelio, Papyrus, Rational Rhapsody, StarUML, Umbrello UML, Visual Paradigm
Information	Data flow	Activity, Class, Component, Composite Structure, Object, Package, Profile	
	Data structure	Class, Component, Composite Structure, Deployment, Object, Package, Profile	
	Data life-cycle	Activity, Profile, Sequence/Communication, State, Timing	
Concurrency	Concurrency structure	Class, Component, Composite Structure, Deployment, Object, Package, Profile	
	Mapping between functional and concurrent components	Component, Composite Structure, Deployment, Package, Profile	
Development	Software modules structure	Class, Component, Composite Structure, Deployment, Object, Package, Profile	
	Software code structure	Class, Component, Composite Structure, Deployment, Object, Package, Profile	
	Software build process	Activity, Class, Component, Composite Structure, Deployment, Object, Package, Profile, Sequence/Communication, State	
	Software release process	Activity, Class, Component, Composite Structure, Deployment, Object, Package, Profile, Sequence/Communication, State	
Deployment	Physical structure	Class, Component, Composite Structure, Deployment, Object, Package, Profile	
	Mapping between functional and physical components	Composite Structure, Deployment, Package, Profile	
Operational	System Installation	Activity, Class, Composite Structure, Component, Deployment, Object, Package, Profile	
	System administration	Activity, Class, Component, Composite Structure, Deployment, Object, Package, Profile, Sequence/Communication, State, Timing, Use-case	
	System configuration	Activity, Class, Component, Composite Structure, Deployment, Object, Package, Profile, Sequence/Communication, State, Timing, Use case	
	System support	Activity, Class, Component, Composite Structure, Deployment, Object, Package, Profile, Sequence/Communication, State, Timing, Use case	
	Systen migration	Activity, Class, Component, Composite Structure, Deployment, Object, Package, Profile, Sequence/Communication, State, Timing, Use case	

2.1. Similar viewpoint frameworks

While we used Rozanski et al.'s viewpoint framework for our survey study as discussed in Section 1.1, many similar frameworks have been existing that offer different considerations of the software architecture viewpoints.[2] Some of those viewpoint frameworks even focus on particular domains (e.g., embedded systems, weapon systems, air-force IT systems, distributed systems, real-time systems, etc.). Our main motivation for Rozanski et al.'s viewpoint framework is essentially to do with their relatively wide scope that can safely be understood and used by any practitioners who work on any types of information systems (i.e., any computer systems that perform some business operations). Indeed, Rozanski et al. got highly inspired from the generic IEEE 1471 standard on architectural description and considered IEEE's separations on the viewpoint and view concepts and their understanding of the view models. Also, Rozanski et al.'s main focus is on the needs of practitioners, who are actually the contributors of our survey. Indeed, in their book, Rozanski et al. guides the architects on putting all the viewpoint descriptions into practice. Table 2 shows the mapping between Rozanski et al.'s viewpoints framework with some pioneering viewpoint frameworks that we discussed in Section 1. Apparently, while those approaches also support the functional, concurrency, and development viewpoints, not all of them actually support the other viewpoints proposed by Rozanski et al. Indeed, the operational concerns of software systems are not in the scope of any of the viewpoint frameworks. The information viewpoint is also ignored by Kruchten's and Soni et al's frameworks. Soni et al. further ignores the deployment viewpoint. Clements et al.'s component & connector (C&C) viewpoint essentially encompasses Rozanski et al.'s multiple viewpoints and thus hinders the modularity which may cause difficulties for the model understandability and analysis in C&C. Lastly, Garland et al.'s approach offers a large set of viewpoints that essentially correspond to Rozanski et al.'s viewpoint model types which are modularised into the viewpoints. However, using such a large set of viewpoints may cause the practitioners to end up with an unnecessarily complex software architecture design that is difficult to understand and trace, and further requires ensuring the consistencies between many different viewpoints and thus hinder the manageability.

2.2. UML and some analytical studies on UML

Since UML is used by many practitioners in industries, UML has been extended by many approaches via UML's profiling mechanism for adapting UML to new domains or platforms. These include Agent UML [20] for multi-agent systems, Secure UML [21] and UMLSec [22] for specifying the security aspects of distributed systems, Context UML [23] for context-aware systems, UML-RT [24] for real-time embedded systems, Flex UML [25] for process-aware information systems, etc. The OMG organisation that standardised UML also proposed some UML profiles such as SoaML [26] for service-oriented architectures, SPEM [27] for software process models, and SysML [28] for systems engineering. How-

[2] The list of viewpoint frameworks can be accessed via the following link: http://www.iso-architecture.org/42010/afs/frameworks-table.html.

Table 2
The mapping between Rozanski et al.'s viewpoint framework and other frameworks.

Viewpoint frameworks	Functional	Information	Concurrency	Development	Deployment	Operational
Kruchten [4]	Logical		Process	Development	Physical	
Soni et al. [5]	Conceptual Architecture		Execution Architecture	Code, Module		
Clements et al. [6]	Component & Connector	Component & Connector	Component & Connector	Module	Alloc-ation	
Garland et al. [7]	Component and Component Interaction	Logical Data, Data Model, Transaction	Process, Proc. State	Layered Subsys., Subsys. Interface Dependency	Deploy-ment and Physical Data	

ever, none of these works consider the practical use of UML for the multiple-viewpoints modeling and instead focus essentially on adapting UML for a particular domain and their needs. Indeed, most of these languages are all centered around the logical and behavior viewpoints and do not provide support for any other viewpoints. The literature includes some analytical studies that compare UML with some other languages (e.g., architecture description languages) [29,30] and also the UML-based languages with each other [31–33]. However, again, the support for multiple viewpoints is ignored in those comparisons and instead the languages' support for particular domains (e.g., embedded systems design) or particular aspects (e.g., mobility and software process modeling) are focused. Recently, Ozkaya [9] has analyzed more than 120 different architectural languages for a number of requirements including the different viewpoints considered in this study. While Ozkaya's analysis results aid in understanding which architectural languages (including some UML-based languages) support which viewpoints, it does not reveal the practitioners' preferences among different viewpoints and give any clue about which types of UML diagrams are more preferable for describing different types of models in each viewpoint.

In the rest of this section, the surveys that have been conducted on practitioners with the focus on their UML usages are discussed.

2.2.1. Surveys on UML

The literature includes many survey studies that have been conducted on UML. However, these surveys essentially focus on either (*i*) the usage frequencies of UML diagrams, (*ii*) practitioners' evaluation of UML for some properties (e.g., code generation and user-friendliness, and notational complexity), (*iii*) practitioners' motivation/demotivation for UML, and (*iv*) practitioners' experience with UML for some software development aspects (e.g., maintenance). Unlike our survey discussed in this paper, none of the existing surveys focus on the software architecture viewpoints in a comprehensive and systematic way and practitioners' UML usage for different viewpoints. Indeed, to the best of our knowledge, our survey is the first empirical study on the software architecture viewpoints that consider various viewpoints for the software design and development activities (e.g., Rozanski et al. [19]'s viewpoints), the different types software models that can be specified for each viewpoint, and practitioners' experiences on the UML diagrams for each viewpoint model.

Petre [34] interviewed 50 software developers who work in 50 different companies to understand the different usage patterns for UML. Petre's result show that most of the practitioners do not even use UML, while the rest consider using UML informally without following any formal procedures. This includes the informal purposes, such as personal use, early modeling, and prototyping. A few developers also perform the automated code generation from their UML specifications.

Nugroho et al. [35] surveyed among 80 software developers who work in companies from different industries in Netherlands. Nugroho et al. prepared 20 different questions for the purpose of understanding practitioners' opinions about (*i*) the completeness of the UML models, (*ii*) the preciseness (i.e., the level of details) of the UML models, (*iii*) transforming models into implementation, (*iv*) UML's productivity for different stages of software development.

Wrycza et al. [36] surveyed among 180 different students who use UML in their university studies. Wrycza et al. asked the students 17 different questions for understanding (*i*) their opinions about UML's complexity, the diagrams with complex notations, and the usefulness and user-friendliness of UML diagrams, (*ii*) whether they find the existing UML diagrams adequate for their software design, (*iii*) their interest towards the source-code generation from the UML diagrams, and (*iv*) their diagram choices for the dynamic view specifications.

Lange et al. [37] surveyed among 80 different architects who work in different domains. In their survey, Lange et al. seek to understand the architects' interest towards different viewpoints that IBM's Rational Rose modeling tool support in their environment. These are the use-case, logical, component, deployment, and scenario viewpoints. According to the results, while use-case and logical viewpoints are the top used ones with UML, the deployment viewpoint is the least used one.

Osman et al. [38] surveyed among 32 software developers who are forced to answer 13 different questions. Osman et al. intended in their survey to learn what types of information that practitioners wish to specify with UML's class diagram and what information they do not need in the class diagram. By doing so, UML's class diagram can be simplified by abstracting out the unnecessary details.

Fernandez-Saez et al. [39] surveyed among 178 practitioners that work in 12 different countries. Fernandez-Saez et al.'s survey includes 28 different questions with the purpose of understanding to what extent practitioners use UML for documenting the software maintenance.

Dobing et al. [40] surveyed among 299 practitioners from the IT industry. Dobing et al. focused on 6 different UML diagrams, which are class diagram, use case diagram, sequence diagram, activity diagram, state-chart diagram, and collaboration (aka communication) diagram. In their survey, Dobing et al. aimed at understanding practitioners' usage ratio for those 6 UML diagrams and the correlations between the use of different diagrams. Dobing et al. also considered any correlations between the project size and the UML diagrams and the UML diagram(s) that are more attractive to the clients who are involved in the system developments.

Grossman et al. [41] surveyed among 150 practitioners from different profiles (i.e., countries, job titles, experiences, etc.) and prompt them to answer 32 different questions that address 8 key properties for evaluating practitioners' experience on UML (i.e., how useful UML is to them). These properties are (*i*) right data, (*ii*) accuracy, (*iii*) compatibility, (*iv*) flexibility, (*v*) understandability, (*vi*) level of details, (*vii*) training, and (*viii*) ambiguity.

Reggio et al. [42] surveyed among 275 different practitioners from industry and academia who have been asked to answer 32 different questions. Reggio et al. aimed at understanding the usage frequencies of the UML diagrams and how this is affected by the user profiles.

Agner et al. [43] surveyed among 209 developers working in the embedded software domain. Agner et al. asked 20 different questions for the purpose of understanding the embedded developers' experience

with UML. Agner et al. essentially aimed at answering the research questions such as why UML is not (or just partially) used by the developers, whether UML satisfies their needs or not, whether they find UML complex or not, and which UML diagrams are used more than the others for the embedded software design.

3. Research questions

In this survey, we seek to answer the following research questions that aid in achieving the paper goal stated in Section 1.1.

RQ1: Which software architecture viewpoint(s) do practitioners consider in modeling their software architectures? In this question, Rozanski et al.'s viewpoint framework has been considered the following viewpoints have been focused upon in our survey: functional, information, concurrency, deployment, development, and operational. The goal is to determine for each of those viewpoints the frequency of practitioners who model their software architectures in that viewpoint.

RQ2: Which type(s) of software architecture models do practitioners specify in each viewpoint that they are interested in? In this question, we focus on the different types of software models that Rozanski et al. proposed for each of their architecture viewpoints. So, the goal is to determine for each viewpoint model considered the frequency of practitioners who specify that model as part of their software architecture design.

RQ3: Which UML diagram(s) do practitioners prefer to use for specifying the software architecture models in each viewpoint? This question also focuses on Rozanski et al.'s viewpoint model types considered. So, the goal is to determine for each software architecture viewpoint model type which UML diagram(s) are used by practitioners for creating software models in that type as part of their software architecture design.

RQ4: Which UML modeling tool(s) do practitioners prefer to use for modeling their software architectures in each viewpoint? In this question, a set of popular well-known UML modeling tools are focused, including MagicDraw, Visual Paradigm, IBM Rational Rhapsody, Enterprise Architect, Modelio, Obeo UML Designer, and StarUML. The goal here is to determine for each viewpoint considered the UML modeling tool(s) that practitioners prefer to use for performing their modeling activities in that viewpoint.

4. Survey design, execution, and sampling

4.1. Survey design

To design our survey, we have used our knowledge and experiences from our previous surveys that we conducted in the near past [14,17]. Moreover, our survey has been inspired from Rozanski et al.'s seminal book on software architecture viewpoints [19]. Indeed, we prepared a set of survey questions for each architecture viewpoint suggested by Rozanski et al., and each question targets a particular model type that Rozanski et al. proposed to be specified in that viewpoint.

Before publishing our survey, we conducted a pilot study among set of practitioners who have been using UML for several years for the multiple-viewpoints modeling of software systems in diverse industries. We initially contacted 10 different practitioners whom we know from the past projects and who have got 10+ years of experience on UML. We let those 10 practitioners know about the survey motivation, goal, and research questions. 5 practitioners have accepted to participate in the pilot study, and we provided them with the survey questionnaire form. Note here that we interacted with the practitioners either by e-mail or face-to-face meetings depending on their preferences. Each practitioner has been given 7 days to return us their feedback about the following issues: (*i*) the survey structure (i.e., sections, sub-sections, and their order), (*ii*) missing/ambiguous questions and answer lists, (*iii*) the dependencies between the questions, (*iv*) the question types (e.g., tabular, multiple-choice, yes/no, and free-text), (*v*) the expected amount of time to fill in the survey, (*vi*) any supplementary materials needed, and (*vii*) how to reach the potential participants. In case of the conflicting feedback of different practitioners, we conducted further meetings with those practitioners to reach on a consensus. As a result, we managed to get rid of many issues that the practitioners pointed out and fixed the survey questions. Moreover, as many of the practitioners requested, we decided to add a supplementary material to the survey questions to make them more precise. Since we based the survey on the architecture viewpoints proposed by Rozanski et al., we used the book web-site which gives the most precise introductory information about the viewpoints.[3] Rozanski et al. provides in their book web-site a separate link for each viewpoint and describes the viewpoint in terms of its (*i*) definition, (*ii*) concerns, (*iii*) model types, (*iv*) pitfalls, (*v*) stakeholders, and (*vi*) applicability. So, we provided the link for each viewpoint on the relevant survey section, and thus, any practitioners who wish to be further informed about the viewpoints before answering the relevant questions may then click on the link and get all the necessary information.

The survey has been fixed with 35 different questions that target the research questions stated in Section 3. Table 3 shows the survey questions, their features, and the research questions that the survey questions are associated with. So, the survey questions can basically be grouped as (*i*) the profile questions (1–6), (*ii*) the functional viewpoint questions (7–9), (*iii*) the information viewpoint questions (10–14), (*iv*) the concurrency viewpoint questions (15–18), (*v*) the development viewpoint questions (19–24), (*vi*) the deployment viewpoint questions (25–28), and (*vii*) the operational viewpoint questions (29–35). As indicated in Table 1, each viewpoint question group includes a separate question for understanding the UML diagrams used for each model type considered in the corresponding viewpoint and a question for understanding the UML tool used for that viewpoint. Most of the survey questions are multiple-answer questions that flexibly allow the participants to choose any of the given answer(s). Note here that some of the multiple-answer questions offer participants with the option of typing their own answers (i.e., free text). This is essentially the case with some profile questions and the questions that aim to learn the UML tools which practitioners use for each viewpoint modeling. With the free text options, the participants may type (*i*) the name(s) of any tools that they use but are not given in the answer list of the question or (*ii*) any kind of projects/industries/jobs that are not given in the answer list of the profile questions. To analyse the free-text answers, we followed the coding strategy manually [44]. That is, we coded each free-text answer to determine its category that can either be among the existing categories given in the answer list of the question or considered as a new category. If the free-text answer is appropriate for multiple answers in the existing answer list, we counted it for each of those answers. If the free-text answer is not existing, we added it to the answer-list as a new answer of the associated survey question. Some of the participants typed inappropriate answers, such as "university" for the work industry, which we had to omit for the questions. To minimise any biases for coding, we both performed coding individually at different times. Then, we conducted a meeting to discuss and compare our individual suggestions of interpretations for each coding and agreed on the most accurate interpretation of the free-text answers. The other questions are Yes/No questions without any choice for free-text. However, to enhance precision, the Yes/No questions have been presented with the rating answers, which are *always (100%), much of the time (> 70%), often (≥ 50%), sometimes (< 50%)*, and *never (0%)*.

4.2. Survey execution

Our survey has been conducted online via *Google forms*[4] and the survey was available online between March 2018 and June 2018. Given our focus on the practitioners' UML usage for different architecture viewpoints, we target in the survey the practitioners who are involved in the

[3] Rozanski et al.'s book web-site that shares the precise information on their viewpoints: https://www.viewpoints-and-perspectives.info/home/viewpoints/.

[4] https://docs.google.com/forms.

Table 3
Survey questions.

Res. que.	Survey questions	Multiple answer	Free text	Yes/no ques.	Numeric answer
Profile Questions	1- Which country do you work in?	Y	N	N	N
	2- What is (are) your current job position(s)?	Y	Y	N	N
	3- Which industry(ies) do you work in?	Y	Y	N	N
	4- What are the type(s) of software projects that you are involved in?	Y	Y	N	N
	5- How many years of experience do you have in software development?	N	N	N	Y
	6- Do you use UML for modeling software systems from different viewpoints?	N	N	Y	N
RQ1	7- Do you model the functional view(s) of software systems in UML?	N	N	Y	N
RQ2, RQ3	8- Which UML diagram(s) do you use for modeling the functional structure of software systems?	Y	N	N	N
RQ4	9- Which UML modeling tool(s) do you use for modeling the functional views of software systems in UML?	Y	Y	N	N
RQ1	10- Do you model the information view(s) of software systems in UML?	N	N	Y	N
RQ2, RQ3	11- Which UML diagram(s) do you use for modeling the data flow of software systems?	Y	N	N	N
	12- Which UML diagram(s) do you use for modeling the data structure of software systems?	Y	N	N	N
	13- Which UML diagram(s) do you use for modeling the data lifecycle of systems (i.e., how data changes over time)?	Y	N	N	N
RQ4	14- Which UML modeling tool(s) do you use for modeling the information views of software systems in UML?	Y	Y	N	N
RQ1	15- Do you model the concurrency view(s) of software systems in UML?	N	N	Y	N
RQ2, RQ3	16- Which UML diagram(s) do you use for modeling the concurrency structure of software systems (i.e., threads and processes)?	Y	N	N	N
	17- Which UML diagram(s) do you use for mapping the functional components into the concurrency components?	Y	N	N	N
RQ4	18- Which UML modeling tool(s) do you use for modeling the concurrency views of software systems in UML?	Y	Y	N	N
RQ1	19- Do you model the development view(s) of software systems in UML?	N	N	Y	N
RQ2, RQ3	20- Which UML diagram(s) do you use for modeling the structure of software modules?	Y	N	N	N
	21- Which UML diagram(s) do you use for modeling the source-code structures?	Y	N	N	N
	22- Which UML diagram(s) do you use for modeling the software build process?	Y	N	N	N
	23- Which UML diagram(s) do you use for modeling the software release process?	Y	N	N	N
RQ4	24- Which UML modeling tool(s) do you use for modeling the development views of software systems in UML?	Y	Y	N	N
RQ1	25- Do you model the deployment view(s) of software systems in UML?	N	N	Y	N
RQ2, RQ3	26- Which UML diagram(s) do you use for modeling the physical structure of software systems?	Y	N	N	N
	27- Which UML diagram(s) do you use for mapping the functional components into physical components?	Y	N	N	N
RQ4	28- Which UML modeling tool(s) do you use for modeling the deployment views of software systems in UML?	Y	Y	N	N
RQ1	29- Do you model the operational view(s) of software systems in UML?	N	N	Y	N
RQ2, RQ3	30- Which UML diagram(s) do you use for modeling the system installation elements and their dependencies?	Y	N	N	N
	31- Which UML diagram(s) do you use for modeling the system administration issues ?	Y	N	N	N
	32- Which UML diagram(s) do you use for modeling the system configuration issues ?	Y	N	N	N
	33- Which UML diagram(s) do you use for modeling the system support issues ?	Y	N	N	N
	34- Which UML diagram(s) do you use for modeling the system migration issues ?	Y	N	N	N
RQ4	35- Which UML modeling tool(s) do you use for modeling the operational views of software systems in UML?	Y	Y	N	N

software modeling & development in different industries. So, to avoid any biases, we intentionally chose not to send the survey to any practitioners who work in academia with no industry background or those with no software modeling & development experience. However, our observations show that a few academics have filled in the survey, and unfortunately, we had to omit their records during the analysis of the survey results so as to minimise any biases. Also, we did not consider the records from the participants who do not use UML for modeling software architectures from different viewpoints. Indeed, the sixth question of the survey indicated in Table 3 prompts the participants to submit the survey directly without answering the viewpoint modeling questions if they have never used UML for the architectural modeling. So, all these helped us eliminate any potential misleading data for the analysis of the survey results.

To reach as many practitioners from industry as possible, we followed a systematic approach. Initially, we started off with our personal contacts whom we know from the research projects/consultations/conferences/workshops/etc. that we have been involved in the past. We sent an e-mail to 50 different personal contacts. Also, we determined a set of practitioners who have contributed to the well-cited scientific papers about software modeling, software architectures, or UML. We sent an e-mail to 125 such prac-

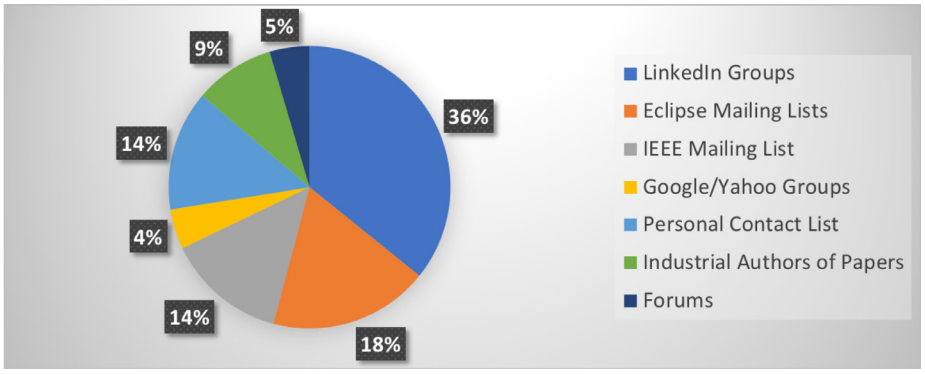


Fig. 2. The distribution of the participants depending on the survey sharing methods.

titioners. Note that for each e-mail sent, we performed snow-balling and kindly requested the practitioners to forward the e-mail to any other practitioners whom they know and consider as the potential participants. Following that, we used the social platforms to spread the survey as much as possible, which was essentially so useful in reaching many practitioners around the world. We initially shared the survey in several related linkedin groups that have 5000+ members and receive many posts/comments actively each day. These include such groups as Software Architects and Enterprise Architects, Software Architecture, The Enterprise Architecture Network, Software As a Service, Software Design Patterns and Architecture, Software Engineering Professionals, Model Driven Architecture, and Software Developer. Moreover, we shared the survey in some google groups that are actively participated by the software engineers. These include UML forum, Istanbul-Coders, Quality Analyst, and MongoDB-User. We also sent the survey e-mail to the active mailing lists in which many software developers who are interested in modeling have been registered with. These include the mailing-list of the IEEE standard for Systems and software engineering – Architecture description (ISO/IEC/IEEE 42010:2011) and the several mailing lists of the Eclipse foundations that are concerned with software modeling (e.g., papyrus-ic, papyrus-sysml, papyrus-jp, mdt-papyrus, emf-dev, and cappella-dev). To further attract interest, we created topics on some active software development forums to kindly invite the forum participants for their contributions to the survey. The forums we used include dev-shed, source-forge, code ranch, and eclipse forums.

As a result, our survey has received 109 different responses from 34 different countries around the world. Fig. 2 shows the rough distribution of the practitioners who contributed to the survey on the methods that can be employed for reaching the survey. So apparently, we got most of the contributions from the linkedin groups (36%), which is followed by the mailing list (Eclipse and IEEE) (32%) and personal contract contributions (14%). The practitioners who contributed to the scientific papers about software modeling just constitute the 9% of the overall participants. The least contributions have been received from the google groups and forums. Note that these data are essentially based on (*i*) the number of positive responses that we received from the participants who used any of those methods for reaching the survey (e.g., reply messages, number of likes, number of comments, etc.), (*ii*) our observations on the response rate during which the particular method has been employed, and (*iii*) our observations on the response rate during which a reminder e-mail/message has been released for a particular method. Therefore, Fig. 2 gives an approximate distribution of the participants, which is expected to be close to the actual distributions.

4.3. Survey sampling

Given the two sampling techniques for selecting participants – probability sampling and non-probability sampling [45], we have used the non-probability sampling method. While probability sampling leads to more precise results, we were not able to use the probability sampling as it requires to access every practitioner of industry who are involved in software development and select them randomly. Indeed, it was not possible for us to reach every practitioner due to the time and budget constraints. Therefore, using the non-probability sampling, we have chosen the participants non-randomly via the methods discussed in Section 4.2.

As we used our personal contacts and the practitioners who contributed to the relevant papers, we essentially applied the convenience sampling of the non-probability sampling technique, in which the practitioners who are easy-to-access have been chosen to be invited. To spread the survey, we also used the online social platforms (i.e., linkedin, google groups, mailing lists, and forums), which essentially let us somewhat mimic the probability sampling. Indeed, many professionals today are the members of linkedin and follow the groups relevant to their expertise. Likewise, several mailing lists on software modeling & development have been actively used by thousands of practitioners today, which enable receiving up-to-date news about the existing or new technologies, any innovations, or share some ideas, papers, and surveys. Forums also play a key role as the discussion platforms on which many practitioners can discuss several relevant issues on software modeling and development. Therefore, each practitioner who uses the social platforms mentioned above can essentially be considered to have an equal chance of accessing the survey. Given thousands of practitioners who actively use social platforms, we were able to reach as many practitioners as possible who are interested in software modeling & development randomly without knowing whom they are and where they work for. However, it should be noted here that it was not possible for us to group the responses of those practitioners and select a sub-set of responses from each group randomly (i.e., a cluster probability sampling [46]). This is due to the fact that in our survey, we did not collect any personal information (e.g., the company name or any other identity data) for grouping as that damages the anonymity of the survey participants and may make the potential participants stop filling the survey form.

Concerning the unit of analysis for the survey sample, our unit is the practitioner who use UML for the software modeling in any industries as discussed in Section 4.2. Each practitioner has participated to the survey individually and anonymously without giving any personal data that may expose their identity.

5. Survey findings and results

5.1. Profiles

5.1.1. The work countries of the participants (Q1)

The survey attracted participants from 34 different countries. These are USA, Colombia, Latvia, Austria, Estonia, Singapore, South Africa, Uruguay, Switzerland, Ukraine, Vietnam, Ghana, Ireland, Australia, Belgium, Brazil, Canada, Finland, France, Costa Rica, Germany, India, Chile, Denmark, Mexico, Italy, Poland, Portugal, Spain, Russia, Sweden, The Netherlands, Turkey, and United Kingdom. Among all those, USA is the top-popular country, followed by India, France, UK, and Turkey.

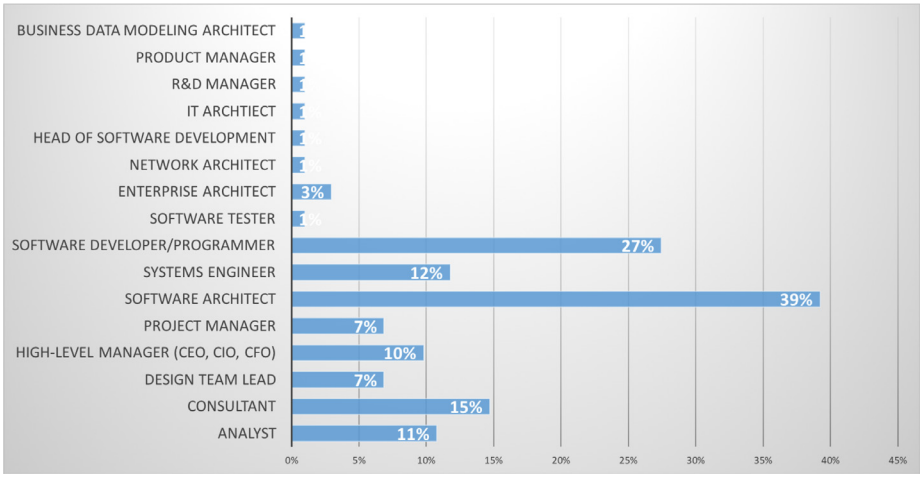


Fig. 3. The current job positions of the participants.

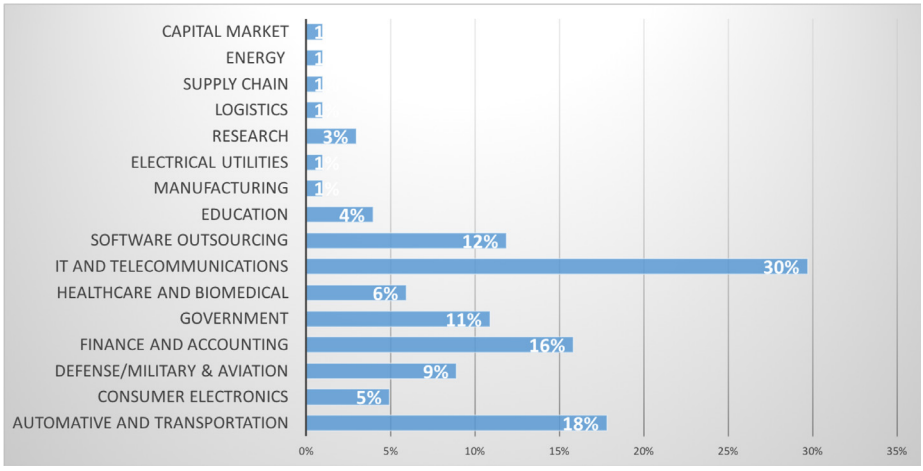


Fig. 4. The work industries for the participants.

5.1.2. The current job positions of the participants (Q2)

As Fig. 3 shows, the software architect is the top-selected job position by participants (39%), followed by the Software Developer/Programmer (27%). While the rest of the job positions are not so popular ($\leq$ 15%), some positions such as IT/network architect, R&D/product manager, and tester are rarely held by the participants.

5.1.3. The work industries of the participants (Q3)

As Fig. 4 shows, IT and Telecommunications is the top-popular work industry, selected by 30% of the participants. Automotive & Transportation and Finance & Accounting are also quite popular industries, selected by 16–18% of the participants. Note that some of the industries such as energy, supply chain, logistics, and electrical utilities are rarely selected by the participants.

5.1.4. The types of software projects that the participants are involved in (Q4)

As Fig. 5 shows, the top-selected software project type is the development of business applications software (54%), followed by the web applications (43%). Some of the other important software project types such as mobile applications, systems software, scientific/engineering applications software, and safety-critical and mission-critical software are just selected by 24–29% of the participants.

5.1.5. The participants' experience on software development (Q5)

As Fig. 6 shows, most of the participants (81%) have 10+ years of experience on software development. While another 16% have at least 2 years of experience, just 3% have no experience at all and thus have been directed to submit the form to avoid any biases.

5.2. UML for different software architecture viewpoints (Q6)

As Fig. 7 shows, most of the participants (88%) use UML in their software architectures modeling from different viewpoints at different levels of frequency. The rest of the participants who never use UML for different viewpoints have been directed to submit the form to avoid any biases.

5.3. Functional viewpoint

Rozanski et al.'s functional viewpoint considers the functional structure models for software systems. A functional structure model is concerned with decomposing software systems into functional components that have some interfaces for interacting with other components. The interfaces of the functional components are linked via the connector elements, which may be as simple as basic links or impose some complex interaction protocols.

5.3.1. Do you model the functional view(s) of software systems in UML? (Q7)

As Fig. 8 shows, most of the participants (96%) model the functional views of their software systems in UML at varying levels of frequency. 4% of the participants who never do so have been directed to the next section (i.e., the information viewpoint).

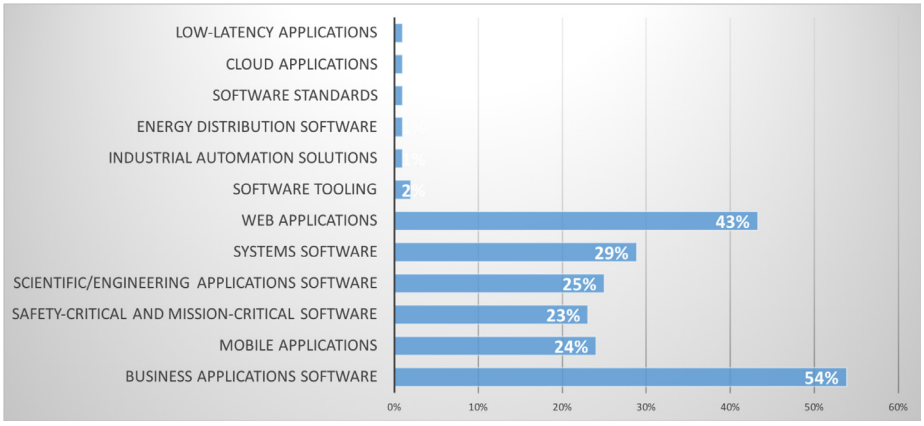


Fig. 5. The types of software projects that the participants are involved in.

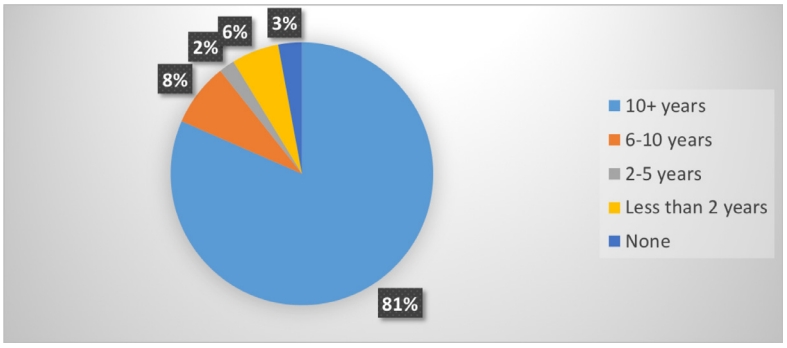


Fig. 6. The years of experiences for the participants.

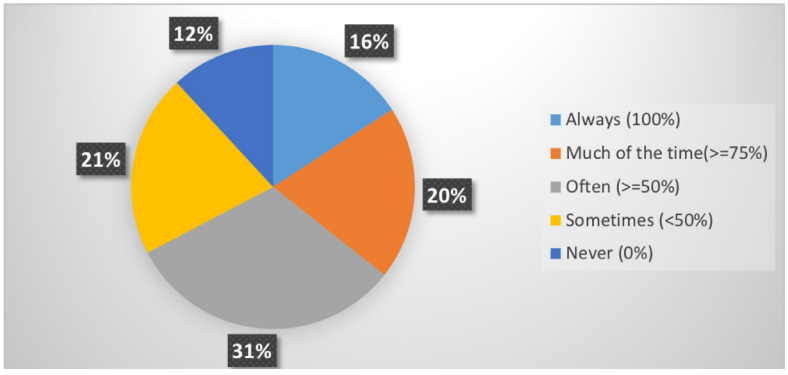


Fig. 7. The frequency of the participants who use UML for modeling their software architectures from multiple viewpoints.

As Table 4 shows, IT & telecommunications is the top-selected industry by the participants who model the functional views in UML, and the top-selected project types are the business applications software and web applications. Concerning their correlations, while the participants from the automotive & transportation and defense/military & aviation industries focus more on the safety-critical and mission-critical software and scientific/engineering applications software, the participants from the finance & accounting, government, healthcare & biomedical, IT & telecommunications, and software outsourcing industries focus on the business applications software and web applications.

5.3.2. Which UML diagram(s) do you use for modeling the functional structure of software systems? (Q8)

As Fig. 9 shows, among the participants who use UML for the functional view modeling, a few of them (8%) never use UML for the functional structure modeling, and another 5% are not interested in the functional structure modeling at all. Concerning the rest, the top-used UML diagram is the class diagram (71%), which is followed by the component and package diagrams (40–58%).

5.3.3. Which UML modeling tool(s) do you use for modeling the functional views of software systems in UML? (Q9)

As Fig. 10 shows, while many different tools are used by the participants, the top-used one is the Enterprise Architect UML modeling tool (53%) and that is followed by the StarUML (16%). Just a few participants (2%) still draw their functional view models by hand. Another interesting outcome is that some participants (14%) specify their UML models for the functional views using the office tools (e.g., MS Office PowerPoint/Word and LibreOffice) that provide no particular support for UML (e.g., UML syntax checking, code generation, UML model versioning, etc.).

Fig. 11 shows the correlation between the work industries of the participants who use some tools for modeling the functional view of their software systems in UML and their tool choices. So apparently, Enterprise Architect is the top-used UML modeling tool in most of the industries - except consumer electronics and healthcare & biomedical. Among the rest of the UML tools, StarUML, MagicDraw, and IBM Rational Rhapsody are used in many of the industries indicated, while their usage ratio is not as high as that of Enterprise Architect. Also, some industries (i.e.,

Table 4
The distribution of the participants who model the functional view depending on the work industries and the software projects involved.

	Business application software	Mobile application	Safety-critical and mission-critical software	Scientific/ engineering applications software	Systems software	Web applications	Total
Automotive and transportation	3 (13%)	0 (0%)	7 (29%)	6 (25%)	7 (29%)	1 (4%)	24 (100%)
Consumer electonics	0 (0%)	0 (0%)	1 (50%)	0 (0%)	1 (50%)	0 (0%)	2 (100%)
Defense/military & aviation	2 (11%)	0 (0%)	7 (39%)	4 (22%)	3 (17%)	2 (11%)	18 (100%)
Finance and accounting	11 (39%)	6 (21%)	0 (0%)	1 (4%)	1 (4%)	9 (32%)	28 (100%)
Government	10 (36%)	2 (7%)	3 (11%)	2 (7%)	4 (14%)	7 (25%)	28 (100%)
Healthcare and biomedical	3 (39%)	1 (12%)	1 (12%)	1 (12%)	0 (0%)	2 (25%)	8 (100%)
IT and telecomunications	17 (31%)	9 (17%)	2 (4%)	5 (9%)	7 (13%)	14 (26%)	24 (100%)
Sofware outsourcing	8 (30%)	6 (22%)	0 (0%)	3 (11%)	3 (11%)	7 (26%)	27 (100%)
TOTAL	54	24	21	22	26	42	

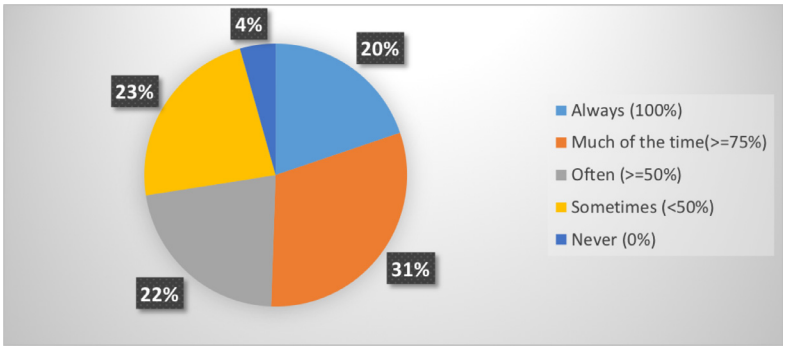


Fig. 8. The frequency of the participants who model the functional view(s) of software systems in UML.

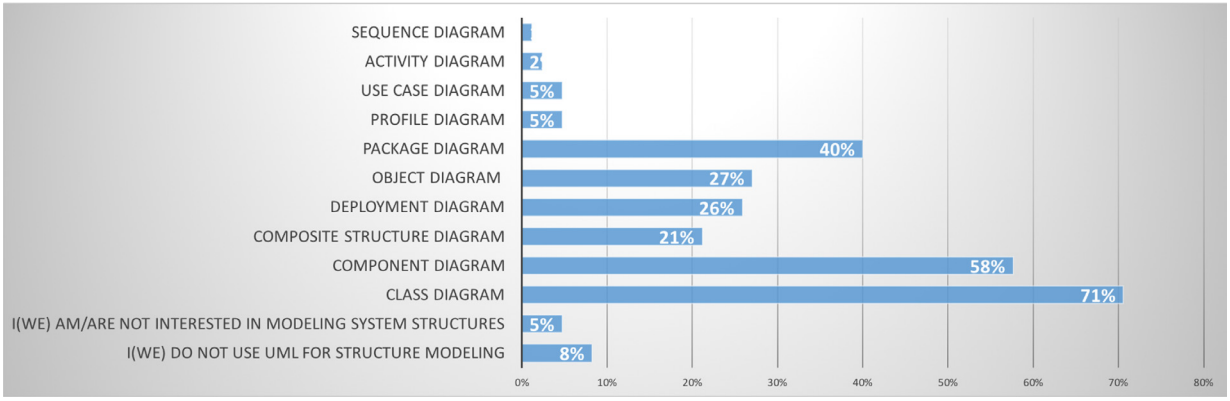


Fig. 9. The UML diagrams used by the participants for modeling the functional structure.

IT & telecommunications, healthcare & biomedical, government, and finance & accounting) use the office tools more than many UML modeling tools despite the office tools lacking in support for the UML modeling.

5.4. Information viewpoint

We consider Rozanski et al.'s three different models for the information viewpoint, which are the static data structure model, information flow model, and information lifecycle model. A static data structure model for a system describes the structure of the system data in terms of the data elements and their relationships with each other. An information flow model describes the data flow between the system components and is concerned with the type, volume, and direction of the data transferred between the components, and how the data transfer occurs (e.g., exchanging flat files or XML messages). An information lifecycle model is concerned with how the data representing the system state transition from one to another due to some events occuring, such as time elapsed and the method-call received/made.

5.4.1. Do you model the information view(s) of software systems in UML? (Q10)

As Fig. 12 shows, almost all the participants model the information views of their software systems in UML (99%) at different levels of frequency. Those who do not so have been directed to the next survey section (i.e., the concurrency viewpoint).

As Table 5 shows, IT & telecommunications is the top-selected industry by the participants who model the information views in UML, and the business applications software is the top-selected project type. Concerning the correlations, the participants from the automotive & transportation and defense/military & aviation industries focus more on the safety-critical & mission-critical software, scientific/engineering applications software and systems software. The participants from rest of the industries considered focus on the business applications software and web applications.

5.4.2. Which UML diagram(s) do you use for modeling the data flow of software systems? (Q11)

As shown in Fig. 13, some participants (17%) do not use UML for modeling the data flow and another 2% are not interested in that

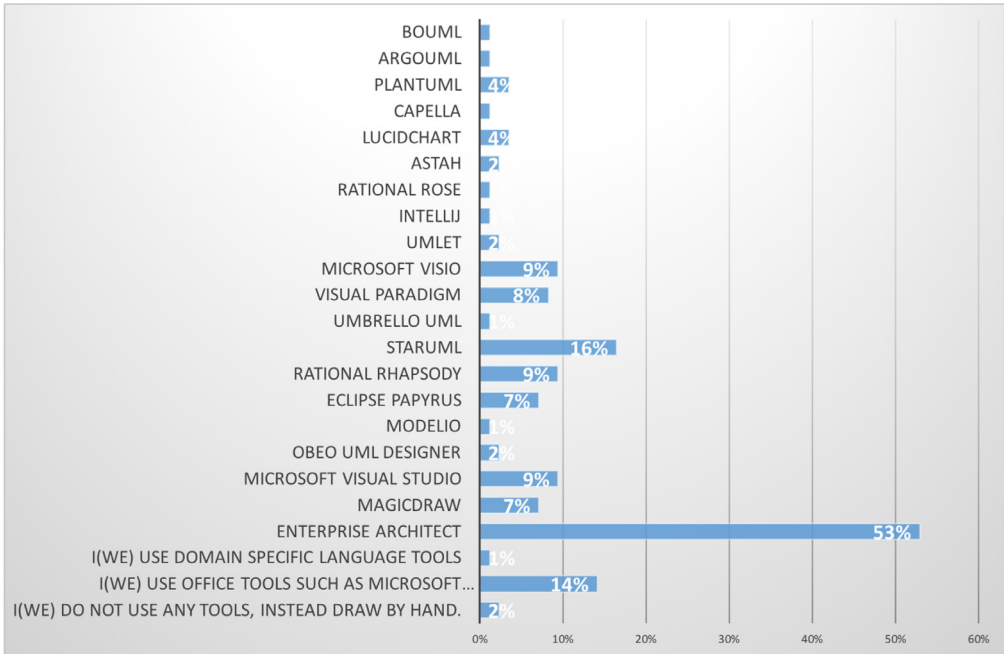


Fig. 10. The UML modeling tools used by the participants for modeling the functional view.

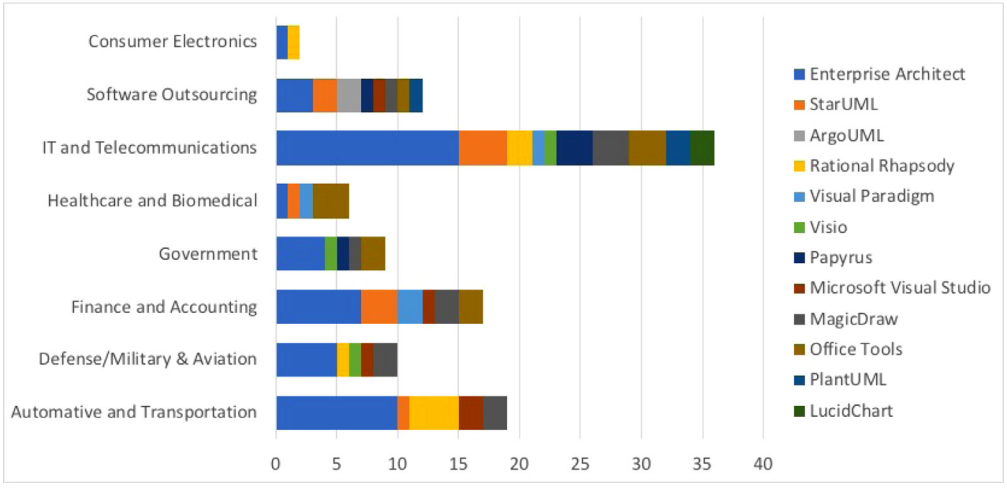


Fig. 11. The correlation between the work industries of the participants and the tool(s) that they use for the functional view modeling in UML.

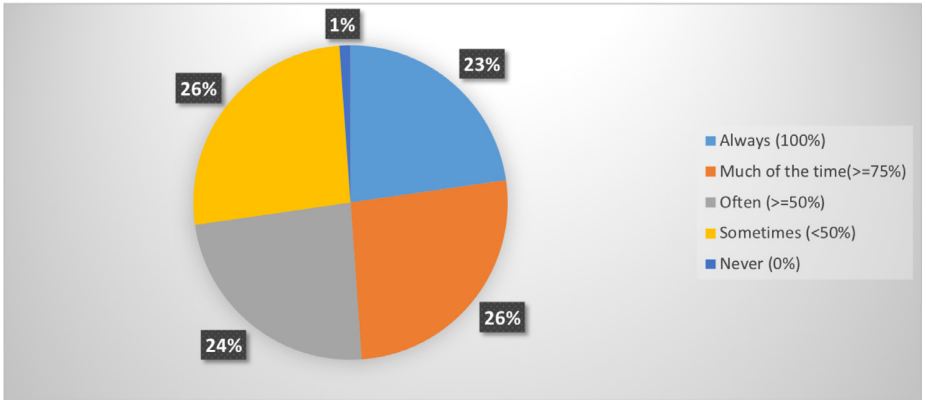


Fig. 12. The frequency of the participants who model the information view(s) of software systems in UML.

Table 5
The distribution of the participants who model the information view depending on the work industries and the software projects involved.

	Business application software	Mobile application	Safety-critical and mission-critical software	Scientific/ engineering applications software	Systems software	Web applications	Total
Automotive and ransportation	3 (13%)	0 (0%)	7 (29%)	6 (25%)	7 (29%)	1 (4%)	24 (100%)
Consumer lectonics	0	0	1 (50%)	0	1 (50%)	0	2 (100%)
Defense/ilitary & viation	2 (10%)	0 (0%)	7 (38%)	4 (21%)	2 (10%)	2 (10%)	19 (100%)
Finance and ccounting	11 (35%)	6 (19%)	0	1 (6%)	3 (10%)	9 (30%)	31 (100%)
Government	10 (36%)	2 (7%)	3 (11%)	2 (7%)	4 (14%)	7 (25%)	28 (100%)
Healthcare and iomedical	3 (40%)	1 (12%)	1 (12%)	1 (12%)	0	2 (24%)	8 (100%)
IT and elecomunications	17 (31%)	9 (17%)	2 (4%)	5 (9%)	7 (13%)	14 (26%)	24 (100%)
Sofware utsourcing	8 (28%)	6 (21%)	0	3 (10%)	5 (17%)	7 (24%)	29 (100%)
TOTAL	54	24	22	23	30	42	

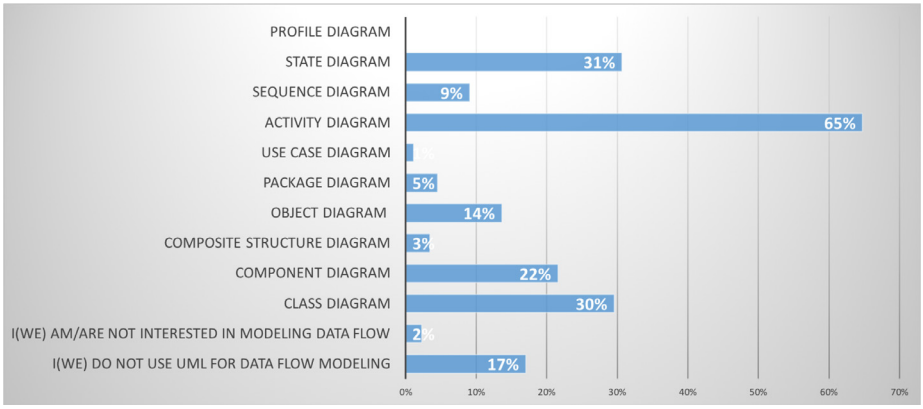


Fig. 13. The UML diagrams used by the participants for modeling the data flow.

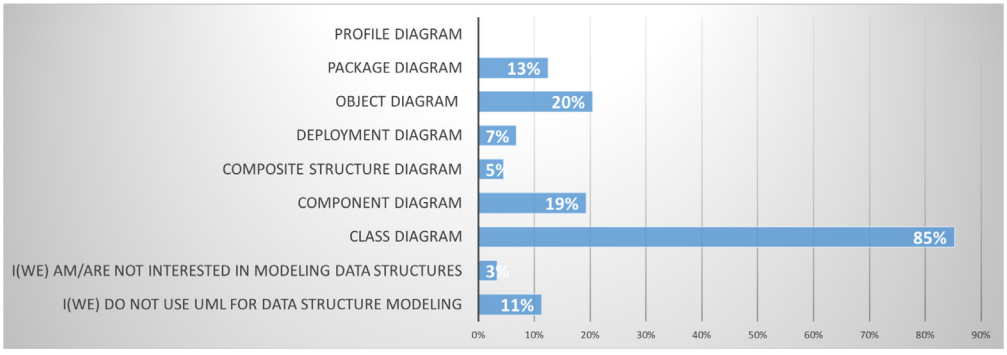


Fig. 14. The UML diagrams used by the participants for modeling the data structure.

at all. Concerning the rest, the top-used UML diagram is the activity diagram (65%), which is followed by the state and class diagrams (30-31%).

5.4.3. Which UML diagram(s) do you use for modeling the data structure of software systems? (Q12)

As shown in Fig. 14, among the participants who use UML for the information view modeling, a few participants (11%) do not use UML for modeling the data structure, and another 3% are not interested in that at all. Concerning the rest, the top-used UML diagram is the class diagram (85%), which is followed by the object and component diagrams (19-20%).

5.4.4. Which UML diagram(s) do you use for modeling the data life-cycle of systems (i.e., how data changes over time)? (Q13)

As Fig. 15 shows, among the participants who use UML for the information view modeling, almost one-third of them (28%) do not use UML for modeling the data life-cycle and another 15% show no interest on the data life-cycle modeling. Concerning the rest, the top-used UML diagrams are the sequence/communication and state diagrams (45–47%), which is followed by the activity diagram (25%).

5.4.5. Which UML modeling tool(s) do you use for modeling the information views of software systems in UML? (Q14)

As Fig. 16 shows, 8% of the participants simply draw their information models on white papers without using any tools. Among the participants using some UML modeling tools, the participants' top choice is Enterprise Architect (45%). Note that the rest of the UML tools are rarely used. As also the case with the functional view modeling, some participants (18%) use the office tools to create the UML models for the information views.

Fig. 17 shows the correlation between the work industries of the participants and their tool choices for the information view modeling in UML. In all the industries (except healthcare and software outsourcing), Enterprise Architect is the top-used UML modeling tool. The other UML modeling tools that are used in many of the industries considered are Visual Studio, MagicDraw, and Visual Paradigm. The office tools are also used in all the industries, except consumer electronics and defense/military & aviation. Indeed, in the software outsourcing industry,

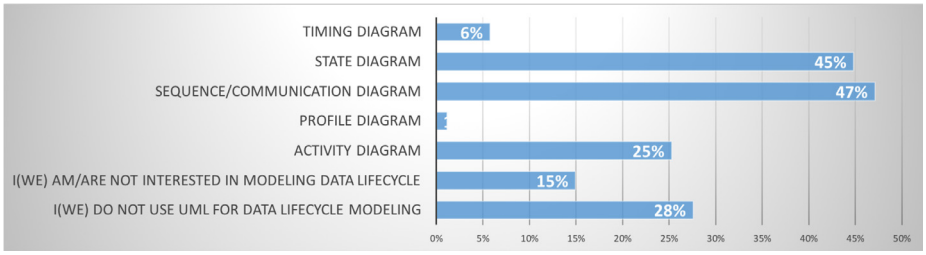


Fig. 15. The UML diagrams used by the participants for modeling the data life-cycle.

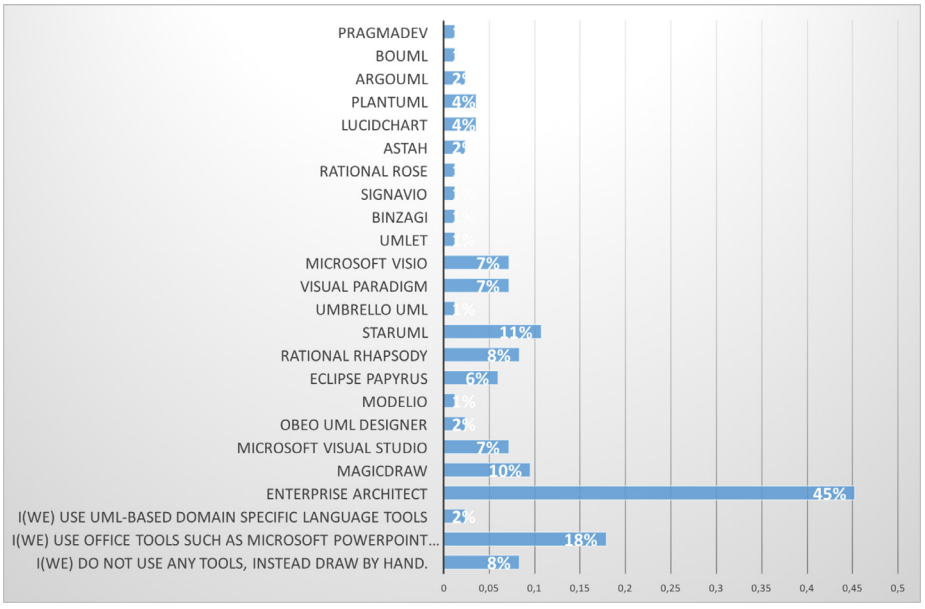


Fig. 16. The UML modeling tools used by the participants for modeling the information view.

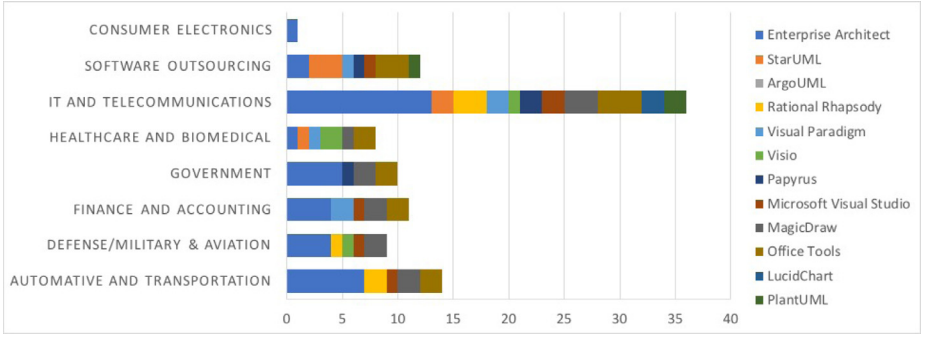


Fig. 17. The correlation between the work industries of the participants and the tool(s) that they use for the information view modeling in UML.

the office tools and StarUML are used even more than Enterprise Architect.

5.5. Concurrency viewpoint

We consider Rozanski et al.'s two different models for the concurrency viewpoint, which are the system-level concurrency model and the state model. A system-level concurrency model describes the concurrency structure of software systems in terms of the runtime execution elements, such as processes and threads, and their concurrent interactions. The system-level concurrency model also addresses the mapping of the functional components into the concurrent components to show which functional components run concurrently within the same process or different processes. A state model is concerned with the state transitions for the concurrency system elements (e.g., threads and processes) and describes how the concurrency elements transition their states depending on any events occuring.

5.5.1. Do you model the concurrency view(s) of software systems in UML? (Q15)

As Fig. 18 shows, two-thirds of the participants (66%) model the concurrency views of their software systems in UML. Those who never model the concurrency views in UML have been directed to the next section (i.e., development viewpoint).

As Table 6 shows, IT & telecommunications and software outsourcing are the top-selected industries by the participants who model the concurrency views in UML, and business applications software and web applications are the top-selected project types. Concerning the correlations, the participants from the automative & transportation and defense/military & aviation industries focus on the safety-critical & mission-critical software, scientific/engineering applications software, and systems software projects. The participants from the rest of the industries focus more on the business applications software and web applications.

Table 6
The distribution of the participants who model the concurrency view depending on the work industries and the software projects involved.

	Business application software	Mobile application	Safety-critical and mission-critical software	Scientific/ engineering applications software	Systems software	Web applications	Total
Automotive and transportation	1 (7%)	0 (0%)	3 (20%)	5 (33%)	6 (40%)	0 (0%)	15 (100%)
Consumer lectonics	0 (0%)	0 (0%)	1 (50%)	0 (0%)	1 (50%)	0 (0%)	2 (100%)
Defense/ilitary & viation	1 (7%)	0 (0%)	6 (40%)	3 (20%)	3 (20%)	2 (13%)	15 (100%)
Finance and ccounting	6 (30%)	5 (25%)	0 (0%)	2 (10%)	1 (5%)	6 (30%)	20 (100%)
Government	3 (44%)	0 (0%)	1 (14%)	1 (14%)	0 (0%)	2 (28%)	7 (100%)
Healthcare and iomedical	1 (25%)	0 (0%)	1 (25%)	0 (0%)	0 (0%)	2 (50%)	4 (100%)
IT and elecomunications	10 (32%)	5 (16%)	2 (6%)	3 (10%)	4 (13%)	7 (23%)	31 (100%)
Sofware utsourcing	7 (26%)	5 (18%)	0 (0%)	3 (11%)	5 (19%)	7 (26%)	27 (100%)
TOTAL	29	15	14	17	20	26	

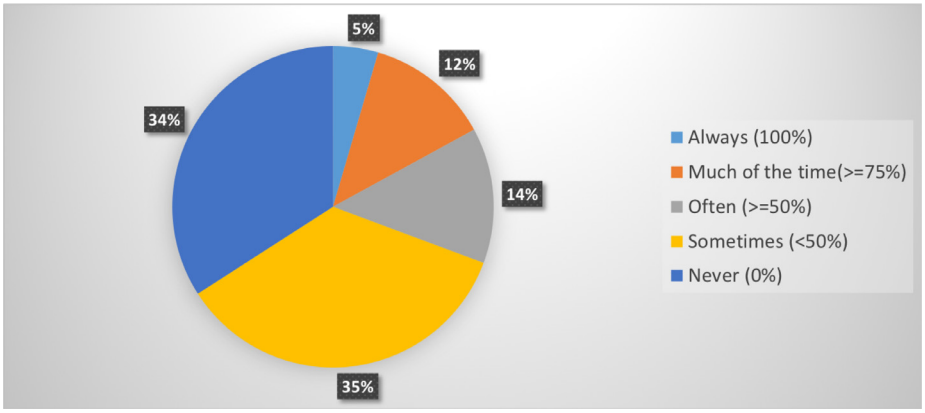


Fig. 18. The frequency of the participants who model the concurrency view(s) of software systems in UML.

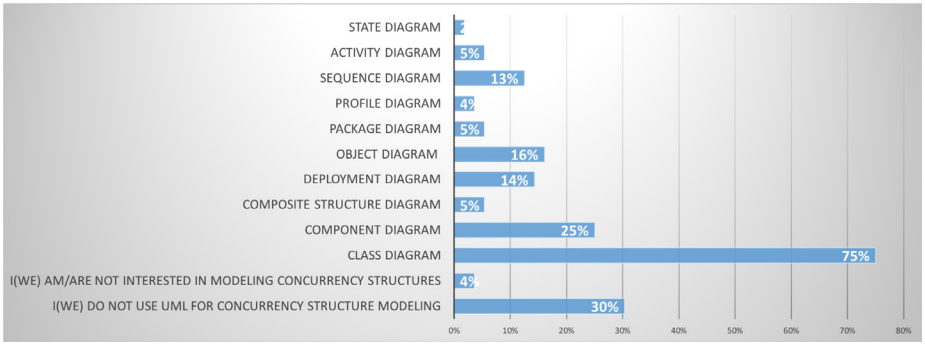


Fig. 19. The UML diagrams used by the participants for modeling the concurrency structure.

5.5.2. Which UML diagram(s) do you use for modeling the concurrency structure of software systems (i.e., threads and processes)? (Q16)

As Fig. 19 shows, among the participants who use UML for the concurrency view modeling, 30% of them never use UML for modeling the concurrency structure and another 4% are not interested in the concurrency structure modeling. Concerning the rest, the top-used UML diagram is the class diagram (75%), followed by the component diagram (25%).

5.5.3. Which UML diagram(s) do you use for mapping the functional components into the concurrency components? (Q17)

As shown in Fig. 20, 35% of the participants do not use UML for mapping the functional components into the concurrency components and another 13% are not interested in the mapping at all. Concerning the rest, the top-used UML diagram is the component diagram (35%), followed by the composite structure and deployment diagrams (19–20%).

5.5.4. Which UML modeling tool(s) do you use for modeling the concurrency views of software systems in UML? (Q18)

As Fig. 21 shows, 25% of the participants draw the UML models for the concurrency views by hand on white papers. Concerning the participants using the UML tools, Enterprise Architect is the most popular UML modeling tools (33%). As is the case with the functional and information viewpoints, the office tools are again used much more than any UML modeling tools (except Enterprise Architect).

Fig. 22 shows the correlation between the work industries of the participants and their tool choices for the concurrency view modeling in UML. Enterprise Architect is again used in all the industries, except healthcare & biomedical. While MagicDraw and Visual Studio are also used by many industries, their usage ratios are not as high as that of Enterprise Architect. Concerning the office tools, again, almost all the industries use them for the UML-based modeling of the concurrency views. The only exception here is the defense/military & aviation industry, who use Enterprise Architect and MagicDraw mainly.

5.6. Development viewpoint

We consider Rozanski et al.'s three models for the development viewpoint, which are the module structure model, the source-code structure model, and the software build & release models. A module structure model for a software system is concerned with structuring the software source files into the software modules that have dependency relationships with each other and grouping those modules into layers. A source-

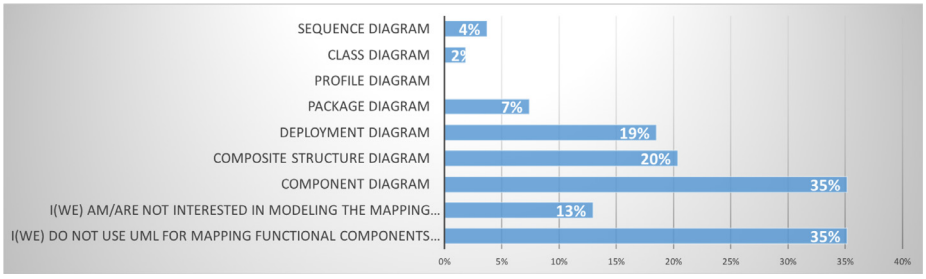


Fig. 20. The UML diagrams used by the participants for modeling the mapping between the functional and concurrency components.

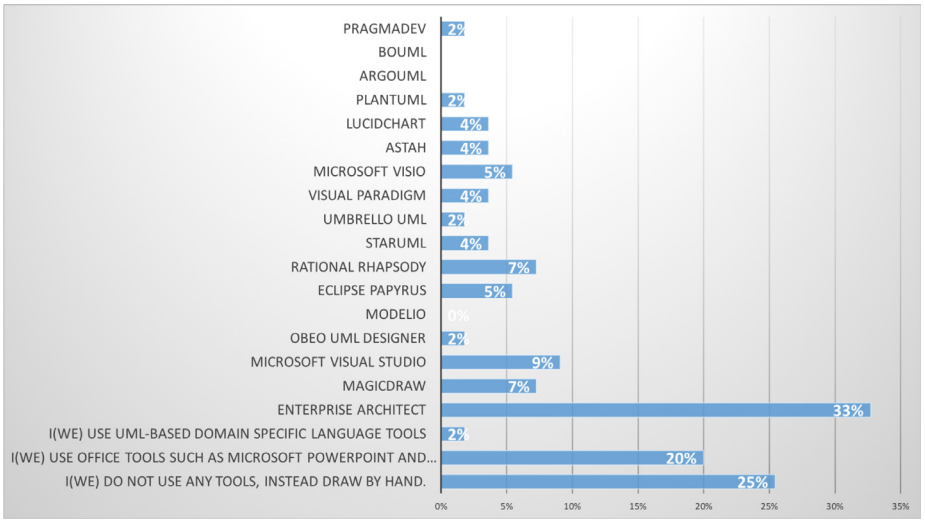


Fig. 21. The UML modeling tools used by the participants for modeling the concurrency view.

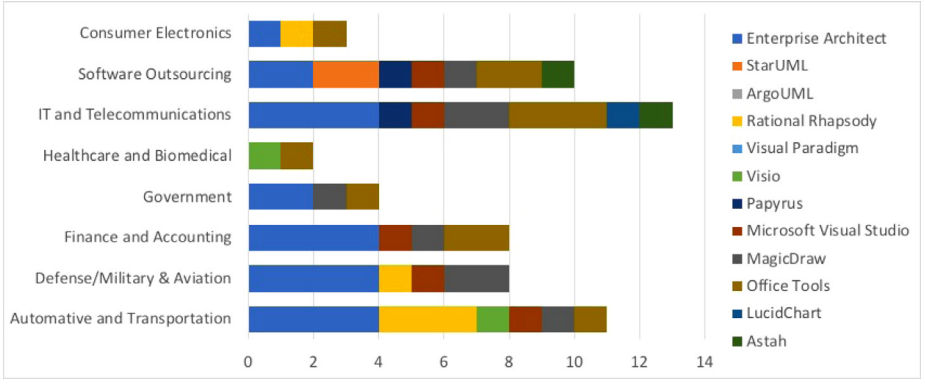


Fig. 22. The correlation between the work industries of the participants and the tool(s) that they use for the concurrency view modeling in UML.

code structure model is concerned with structuring the source code of a software system into the source files. Software build & release process models are concerned with describing the tasks and their relationships that need to be performed in the build & release processes.

5.6.1. Do you model the development view(s) of software systems in UML? (Q19)

As shown in Fig. 23, many of the participants (64%) use UML for modeling the development views at different levels of frequency. Those who do not do so have been directed to the next survey section (i.e., the deployment viewpoint).

As Table 7 shows, IT & telecommunications, software outsourcing, and finance & accounting are the top-selected industries by the participants who model the development views in UML, and business applications software and web applications are the top-selected project types. Concerning the correlations, the participants from the automotive & transportation and defense/military & aviation industries focus on the scientific/engineering applications software and system software. Note that the participants from defense/military & aviation also focus on the safety-critical and mission-critical software projects. The participants who work in the other industries considered focus more on the business applications software and web applications projects.

5.6.2. Which UML diagram(s) do you use for modeling the structure of software modules? (Q20)

As Fig. 24 shows, among the participants who use UML for the development view modeling, a few of them (18%) never use UML here and another 5% are not interested in modeling the software module structures at all. Concerning the rest, the top-used UML diagrams are the component and package diagrams (47%), followed by the deployment diagram (36%).

5.6.3. Which UML diagram(s) do you use for modeling the source-code structures? (Q21)

As Fig. 25 shows, among the participants who use UML for the development view modeling, more than one-third of them (38%) never

Table 7
The distribution of the participants who model the development view depending on the work industries and the software projects involved.

	Business application software	Mobile application	Safety-critical and mission-critical software	Scientific/ engineering applications software	Systems software	Web applications	Total
Automotive and transportation	1 (10%)	0 (0%)	2 (18%)	4 (36%)	4 (36%)	0 (0%)	11 (100%)
Consumer electonics	0 (0%)	0 (0%)	1 (50%)	0 (0%)	1 (50%)	0 (0%)	2 (100%)
Defense/ilitary & viation	1 (10%)	0 (0%)	3 (30%)	3 (30%)	3 (30%)	0 (0%)	10 (100%)
Finance and ccounting	9 (33%)	6 (22%)	0 (0%)	1 (4%)	2 (8%)	9 (33%)	27 (100%)
Government	5 (33%)	7 (7%)	2 (14%)	2 (14%)	2 (14%)	3 (21%)	15 (100%)
Healthcare and iomedical	1 (20%)	1 (20%)	1 (20%)	0 (0%)	0 (0%)	2 (40%)	5 (100%)
IT and elecomunications	14 (34%)	6 (14%)	1 (2%)	3 (7%)	4 (19%)	10 (24%)	42 (100%)
Sofware utsourcing	6 (27%)	4 (18%)	0 (0%)	3 (14%)	4 (18%)	5 (23%)	22 (100%)
TOTAL	29	15	14	14	15	27	

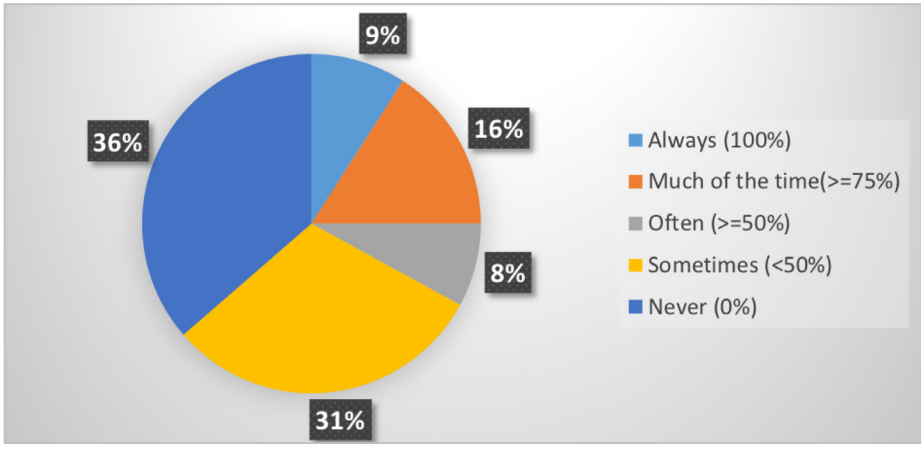


Fig. 23. The frequency of the participants who model the development view(s) of software systems in UML.

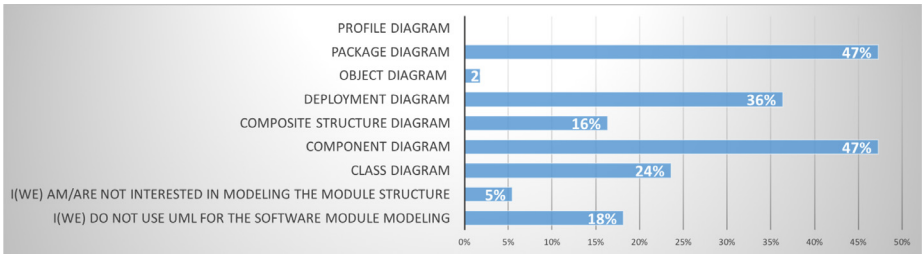


Fig. 24. The UML diagrams used by the participants for modeling the software module structure.

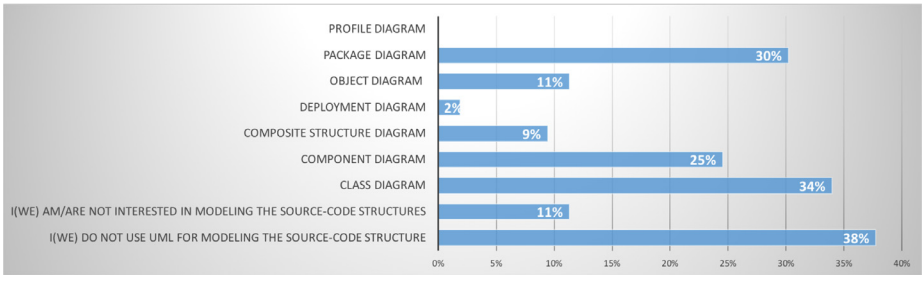


Fig. 25. The UML diagrams used by the participants for modeling the source-code structure.

use UML here and another 11% did not show any interest towards modeling the source-code structure. Concerning the rest, the top used UML diagrams are the class and package diagrams (30-34%), followed by the component diagram (25%).

5.6.4. Which UML diagram(s) do you use for modeling the software build process? (Q22)

As Fig. 26 shows, among the participants who use UML for the development view modeling, more than half of them (55%) never use UML for modeling the software build process, and another 11% are not interested in that at all. Concerning the rest, the top-used UML diagram is the activity diagram (20%), followed by the sequence/communication and deployment diagrams (13-15%).

5.6.5. Which UML diagram(s) do you use for modeling the software release process? (Q23)

As Fig. 27 shows, among the participants who use UML for the development view modeling, half of them never use UML for the modeling of the software release process and another 15% showed no interest at all.

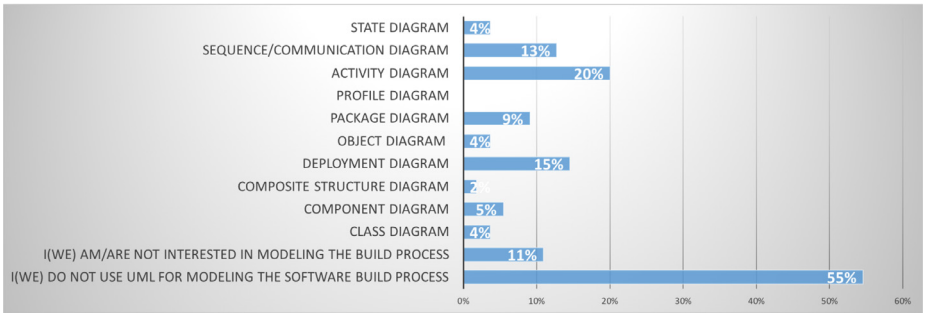


Fig. 26. The UML diagrams used by the participants for modeling the software build process.

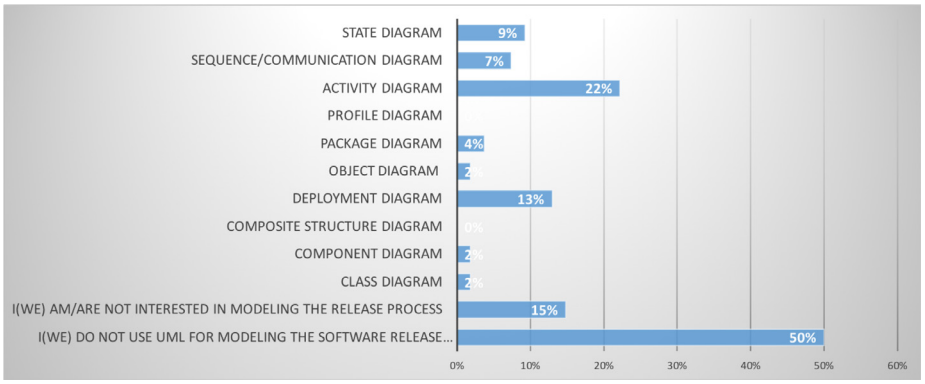


Fig. 27. The UML diagrams used by the participants for modeling the software release process.

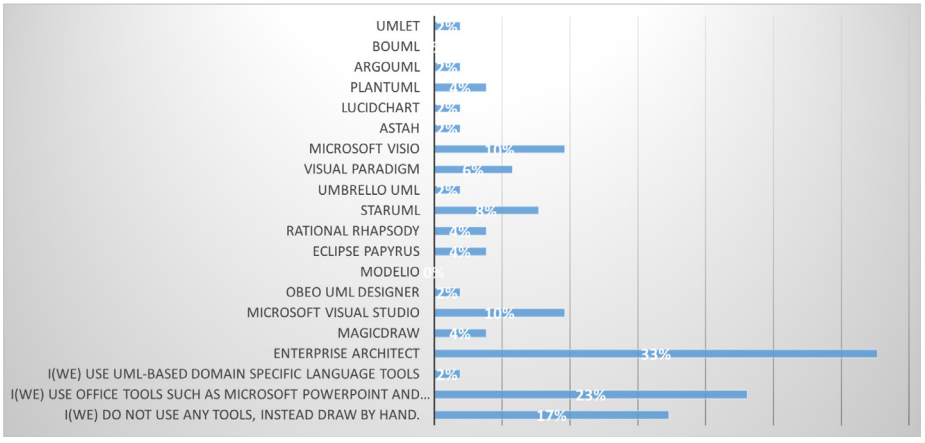


Fig. 28. The UML modeling tools used by the participants for modeling the development view.

Concerning the rest, the top-used UML diagram is the activity diagram (22%), followed by the deployment and state diagrams (9-13%).

5.6.6. Which UML modeling tool(s) do you use for modeling the development views of software systems in UML? (Q24)

As Fig. 28 shows, 17% of the participants do not use any UML tools and instead draw their development view models by hand. Among those using UML tools, Enterprise Architect has been shown the greatest level of interest (33%). Again, some participants (23%) use the office tools so as to specify the UML models for the development views.

The correlation between the work industries and the participants' tool choices is given in Fig. 29. So, the participants of each industry use a varying set of tools for their UML-based development view modeling. However, Enterprise Architect seems to be the top-used tool for many industries including IT & telecommunications, finance & accounting, defense/military & aviation. Note that the healthcare & biomedical industry mainly uses Visio, and the automative & transportation industry uses Rational Rhapsody. Also, the office tools are used in all the industries, while no any UML modeling tools are actually so.

5.7. Deployment viewpoint

We consider Rozanski et al.'s two models for the deployment viewpoint, which are the physical system structure model and deployment model. A physical system structure model is concerned with determining the physical components (e.g., processors, memory, offline storage unit, client and server hardware units), in which the software systems are deployed, and their physical connections (e.g., ethernet and wireless connections). A deployment model is concerned with the mapping between the functional and physical components.

5.7.1. Do you model the deployment view(s) of software systems in UML? (Q25)

As shown in Fig. 30, most of the participants (75%) model the deployment view of their software systems in UML at different levels of frequency. Those who never do so have been directed to the next section of the survey (i.e., operational viewpoint).

As Table 8 shows, IT & telecommunications is the top-selected industry by the participants who model the deployment views in UML,

Table 8
The distribution of the participants who model the deployment view depending on the work industries and the software projects involved.

	Business application software	Mobile application	Safety-critical and mission-critical software	Scientific/ engineering applications software	Systems software	Web applications	Total
Automotive and transportation	3 (17%)	0 (0%)	6 (33%)	4 (22%)	4 (22%)	1 (6%)	18 (100%)
Consumer lectonics	0 (0%)	0 (0%)	1 (50%)	0 (0%)	1 (50%)	0 (0%)	2 (100%)
Defense/ilitary & viation	1 (7%)	0 (0%)	6 (44%)	3 (21%)	3 (21%)	1 (7%)	14 (100%)
Finance and ccounting	9 (32%)	6 (22%)	0 (0%)	2 (7%)	2 (7%)	9 (32%)	28 (100%)
Government	5 (37%)	1 (7%)	1 (7%)	1 (7%)	2 (14%)	4 (28%)	14 (100%)
Healthcare and iomedical	2 (40%)	0 (0%)	1 (20%)	0 (0%)	0 (0%)	2 (40%)	5 (100%)
IT and elecomunications	14 (32%)	7 (16%)	2 (4%)	3 (7%)	7 (16%)	11 (25%)	44 (100%)
Sofware utsourcing	6 (23%)	6 (23%)	0 (0%)	2 (8%)	5 (19%)	7 (27%)	26 (100%)
TOTAL	40	21	18	13	23	35	

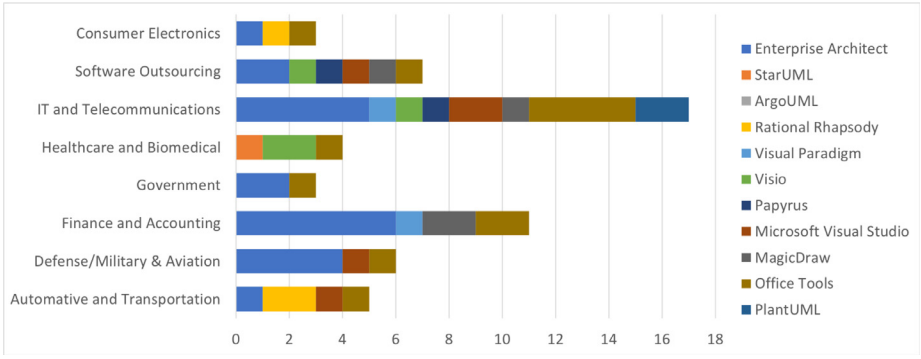


Fig. 29. The correlation between the work industries of the participants and the tool(s) that they use for the development view modeling in UML.

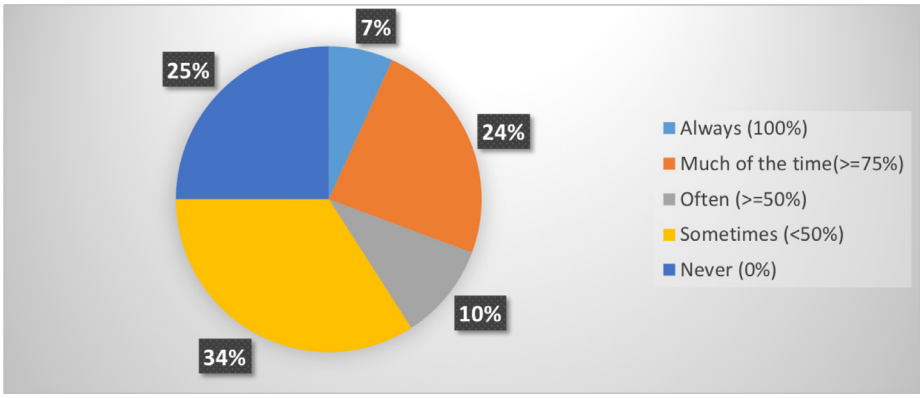


Fig. 30. The frequency of the participants who model the deployment view(s) of software systems in UML.

and, business applications software and web applications are the top-selected project types. Concerning the correlations, the participants from the automotive & transportation and defense/military & aviation industries focus on the safety-critical & mission-critical software, scientific/engineering applications software, and systems software projects. The participants from the rest of the industries focus essentially on the business applications software and web applications. Note that the participants from finance & accounting and software outsourcing are interested in the mobile applications development too.

5.7.2. Which UML diagram(s) do you use for modeling the physical structure of software systems? (Q26)

As shown in Fig. 31, among the participants who use UML for the deployment view modeling, a few of them (18%) do not use UML for modeling the physical structure. Concerning the rest, the top-used UML diagram is the deployment diagram (71%), followed by the component diagram (29%).

5.7.3. Which UML diagram(s) do you use for mapping the functional components into physical components? (Q27)

As Fig. 32 shows, among the participants who use UML for the deployment view modeling, almost one-third of them (31%) do not use UML for the mapping herein and another 8% are not interested in that at all. Concerning the rest, the top-used UML diagram is the deployment diagram (53%), followed by the package and composite structure diagrams (19-23%).

5.7.4. Which UML modeling tool(s) do you use for modeling the deployment views of software systems in UML? (Q28)

As shown in Fig. 33, a few participants (9%) specify their UML models for the deployment views by hand without any tool support. Concerning the participants using UML tools, the top-used modeling tool is Enterprise Architect (45%). Also, some participants (20%) use the office tools to specify their UML models herein.

Fig. 34 shows the correlation between the participants' work industries and the tool they use for the UML-based modeling of the deployment views. Enterprise Architect is again the top-used UML tool for most of the industries. The exceptions herein are the software outsourcing

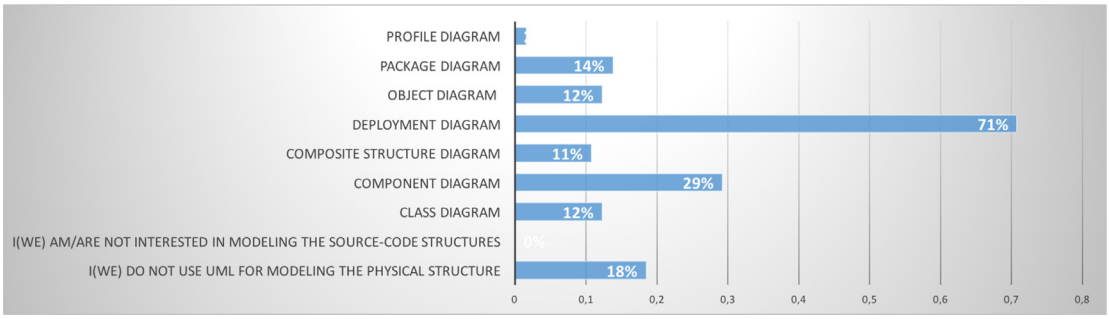


Fig. 31. The UML diagrams used by the participants for modeling the physical structure.

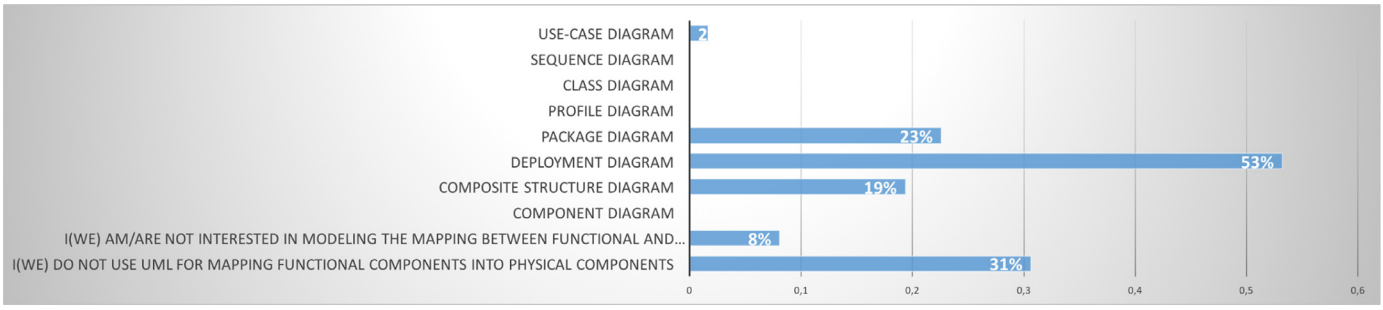


Fig. 32. The UML diagrams used by the participants for modeling the mapping between the functional and physical components.

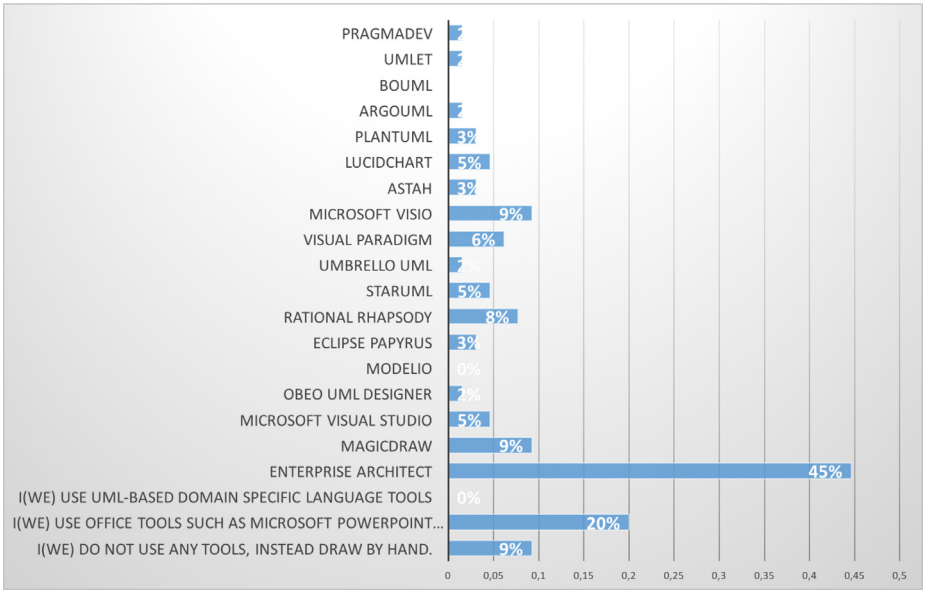


Fig. 33. The UML modeling tools used by the participants for modeling the deployment view.

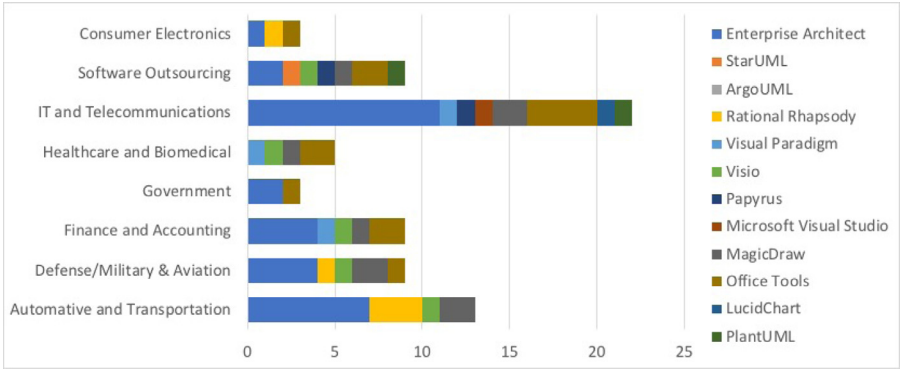


Fig. 34. The correlation between the work industries of the participants and the tool(s) that they use for the deployment view modeling in UML.

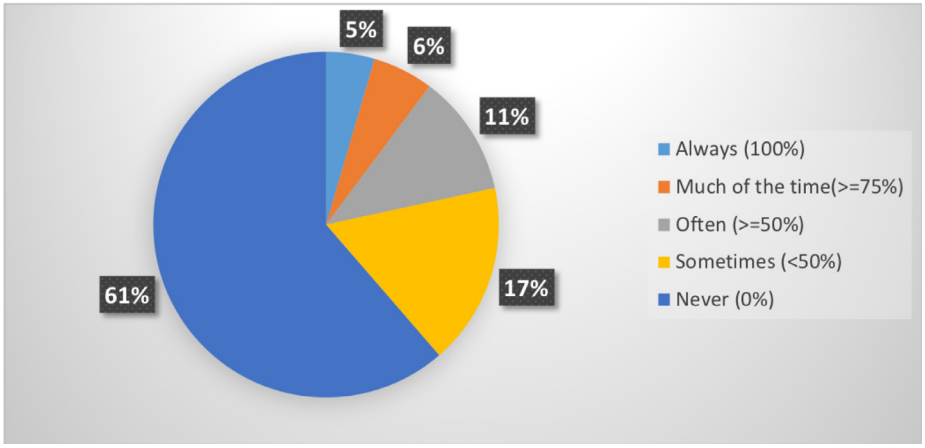


Fig. 35. The frequency of the participants who model the operational view(s) of software systems in UML.

and healthcare industries where the office tools seem to be more popular. MagicDraw and Visio are also used in most of the industries, while their usage ratios are not as high as that of Enterprise Architect.

5.8. Operational viewpoint

We consider Rozanski et al.'s six different models for the operational viewpoint, which are the installation model, migration model, configuration management model, administration model, and support model. An installation model describes the software elements to be installed/upgraded, any dependencies that may exist between different elements, any system constraints, or undo plans in case of the installation errors. A migration model describes any strategies for (*i*) migrating data from some other existing systems, (*ii*) synchronizing some old systems with the new system, or (*iii*) reverting back to the old system in case of any problems. A configuration management model describes the groups of configuration elements (e.g., database management, operating system, and application server configurations) and their dependencies, any strategies for assigning configuration values, and any strategies for performing the configuration changes. An administration model describes the facilities for system monitoring and control, any administrative tasks and their dependencies, and any conditions for errors that administrators need to detect and recover. Lastly, a support model describes the stakeholder groups that need to be supported, the incidents that require support, and the support providers and their coordinations for complex incidents.

5.8.1. Do you model the operational view(s) of software systems in UML? (Q29)

As shown in Fig. 35, the participants showed the least interest for the UML-based modeling of the operational views. Indeed, more than half of the participants (61%) never use UML herein, and they all have been directed to submit the survey without answering the operational viewpoint questions.

As Table 9 shows, finance & accounting and IT & telecommunications are the top-selected industries by the participants who model the operational views in UML, and business applications software and web applications are the top-selected project types. Concerning the correlations, the participants from the automative and transportation industry focus on the business applications software and scientific/engineering applications software. The participants from the defense/military & aviation industry focus on the safety-critical & mission-critical software, scientific/engineering applications software, and systems software. The participants from the finance & accounting industry focuses on business applications software and mobile applications. The participants from the rest of the industries focus essentially on the business applications software and web applications. Note that the participants from software outsourcing are interested in the mobile applications development too.

5.8.2. Which UML diagram(s) do you use for modeling the system installation elements and their dependencies? (Q30)

As Fig. 36 shows, among the participants who use UML for the operational view modeling, more than one-third of them (36%) never use UML herein. Concerning the rest, the top-used UML diagram is the component diagram (39%), followed by the deployment diagram (33%).

5.8.3. Which UML diagram(s) do you use for modeling the system administration issues (e.g., required routine tasks, fault handling, monitoring facilities, etc.)? (Q31)

As Fig. 37 shows, among the participants who use UML for the operational view modeling, almost half of them (45%) never use UML for modeling the system administration issues, and another 3% are not interested in at all. Concerning the rest, the top-used UML diagram is the activity diagram (36%), followed by use case diagram (30%).

5.8.4. Which UML diagram(s) do you use for modeling the system configuration issues (e.g., identifying the configuration groups and their dependencies and any strategies for configuration changes)? (Q32)

As Fig. 38 shows, among the participants who use UML for the operational view modeling, mre than half of them (52%) do not use UML for modeling the system configuration issues (52%) and another 6% are not even interested in modeling the system configuration issues. Concerning the rest, the top-used UML diagram is the component diagram (21%), followed by the use case, activity, deployment, and class diagrams (12%).

5.8.5. Which UML diagram(s) do you use for modeling the system support issues (e.g., the groups that need support, the types of support incidents, and the support providers and their responsibilities)? (Q33)

As Fig. 39 shows, among the participants who use UML for the operational view modeling, half of them never use UML for modeling the system support issues and another 6% are not interested in that at all. Concerning the rest, the top-used UML diagrams are the class and use case diagrams (16%), followed by the sequence/communication and activity diagrams (13%).

5.8.6. Which UML diagram(s) do you use for modeling the system migration issues (e.g., strategies for migrating information and users, data migration, and information synchronization)? (Q34)

As Fig. 40 shows, among the participants who use UML for the operational view modeling, almost half of them (45%) never use UML herein and another 12% are not interested in modeling the system migration issues at all. Concerning the rest, the top-used UML diagram is the deployment diagram (21%), followed by the activity and sequence/communication diagrams (18%).

Table 9
The distribution of the participants who model the operational view depending on the work industries and the software projects involved.

	Business application software	Mobile application	Safety-critical and mission-critical software	Scientific/ engineering applications software	Systems software	Web applications	Total
Automotive and transportation	2 (25%)	0 (0%)	1 (12%)	3 (39%)	1 (12%)	1 (12%)	8 (100%)
Consumer lectonics	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Defense/ilitary & viation	0 (0%)	0 (0%)	4 (36%)	3 (27%)	3 (27%)	1 (10%)	11 (100%)
Finance and ccounting	6 (28%)	5 (24%)	0 (0%)	2 (10%)	2 (10%)	6 (28%)	21 (100%)
Government	3 (32%)	1 (12%)	1 (12%)	0	1 (12%)	3 (32%)	9 (100%)
Healthcare and iomedical	2 (50%)	0 (0%)	1 (25%)	0 (0%)	0 (0%)	1 (25%)	4 (100%)
IT and elecomunications	8 (38%)	4 (19%)	0 (0%)	2 (10%)	2 (10%)	5 (23%)	21 (100%)
Sofware utsourcing	3 (25%)	3 (25%)	0 (0%)	1 (8%)	1 (8%)	4 (34%)	12 (100%)
TOTAL	24	13	7	10	10	21	

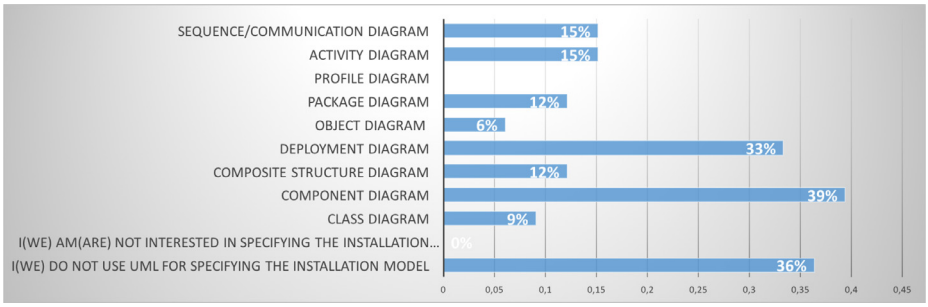


Fig. 36. The UML diagrams used by the participants for modeling the system installation elements.

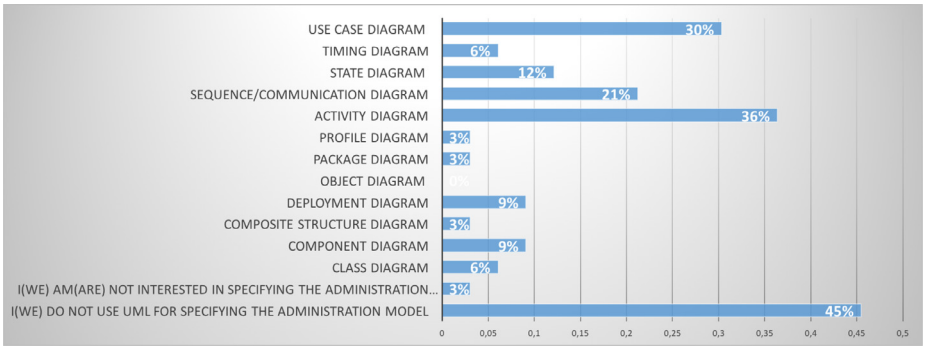


Fig. 37. The UML diagrams used by the participants for modeling the system administration issues.

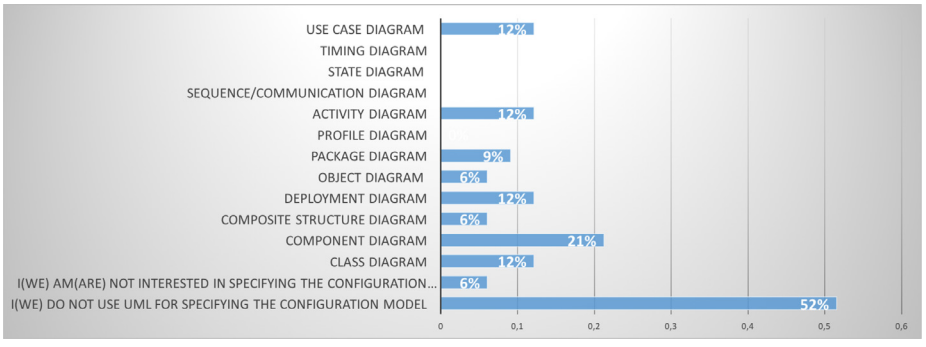


Fig. 38. The UML diagrams used by the participants for modeling the system configuration issues.

5.8.7. Which UML modeling tool(s) do you use for modeling the operational views of software systems in UML? (Q35)

Fig. 41 shows that 23% of the participants never use UML tools and instead draw their models by hand. 16% of the participants use the office tools for modeling the operational views. The top used UML tool is Enterprise Architect (45%). While MagicDraw is used by 13% of the participants, the rest of the UML modeling tools are rarely used by the participants.

Fig. 42 shows the correlation between the work industries and the participants' tool choices for the UML-based operational view modeling. In almost all the industries, more than half of the practitioners use Enterprise Architect. MagicDraw and Obeo UML Designer are the other quite popular UML modeling tools, which are used in many of the industries. Note that the office tools are not that popular this time, which are even totally omitted in some industries such as software outsourcing, healthcare, and defense/military. Also, surprisingly, the top-used

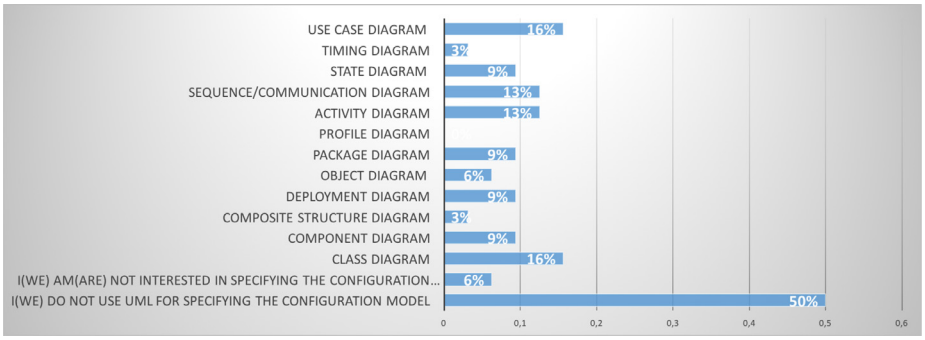


Fig. 39. The UML diagrams used by the participants for modeling the system support issues.

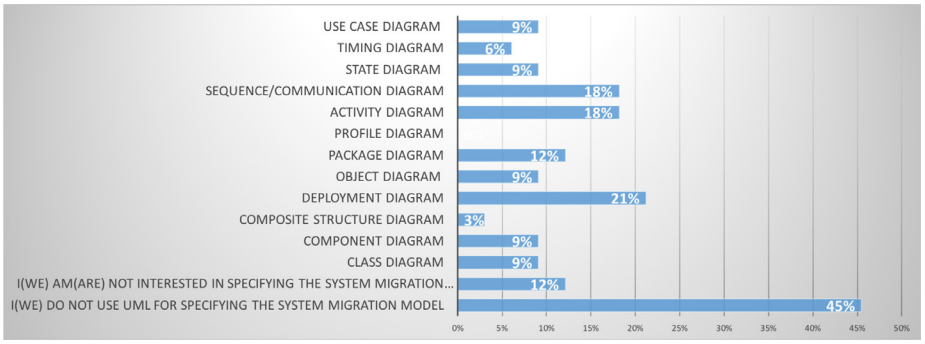


Fig. 40. The UML diagrams used by the participants for modeling the system migration issues.

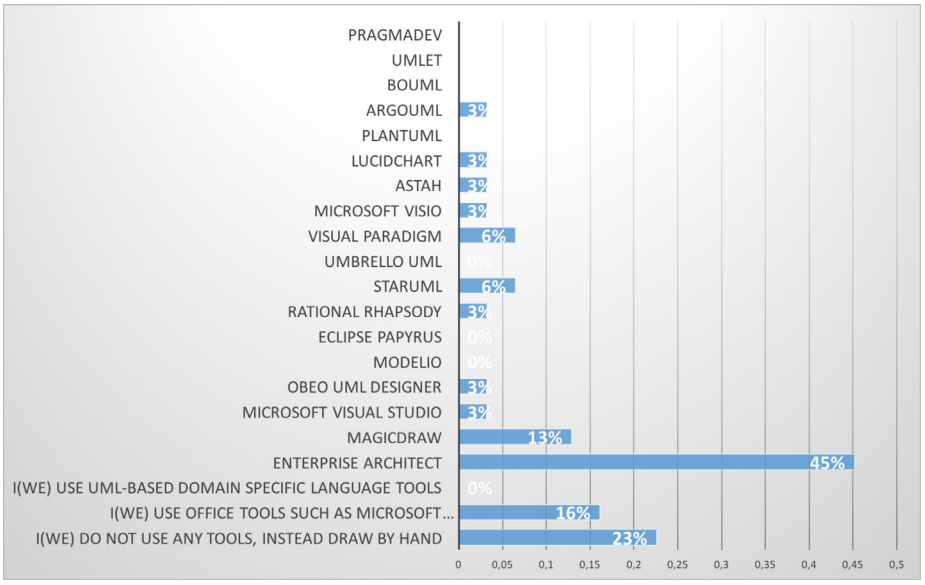


Fig. 41. The UML modeling tools used by the participants for modeling the operational view.

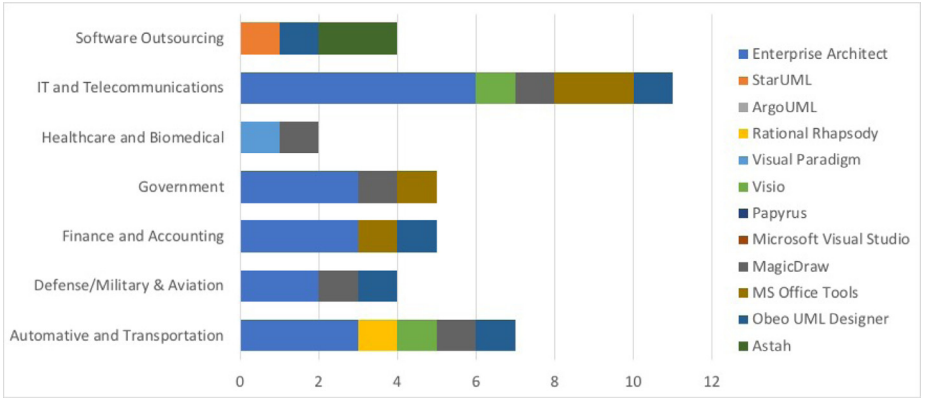


Fig. 42. The correlation between the work industries of the participants and the tool(s) that they use for the operational view modeling in UML.

UML modeling tool for software outsourcing has been observed to be Astah.

6. Discussions

6.1. Summary of findings

While Section 5 discuses the survey results thoroughly, we summarise below some of the key results obtained for each section of the survey. Note that we summarise the top-used UML notations for each viewpoint in terms of the different model types that the viewpoints support rather than indicating the UML diagram usages for the viewpoints in general.

Profile questions. The survey attracted participants from 34 different countries all over the world. USA is the top popular country among the participants. The top-selected job positions are Software Architect (39%) and Software Developer/Programmer position (27%). The top-popular industry is the IT & Telecommunications industry (30%). The top-popular software project types are the Business Applications Software (54%) and Web Applications (43%). Most of the participants (81%) have 10+ years of experience on software development.

Functional viewpoint. 96% of the participants model the functional views of their software systems in UML. The top-used UML diagram for the functional structure modeling is the class diagram (71%).

Information viewpoint. 99% of the participants model the information views of their software systems in UML. The top-used UML diagrams are (*i*) the activity diagram for the data-flow modeling (65%), (*ii*) the class diagram for the data structure modeling (85%), and (*iii*) the sequence/communication and state diagrams for the data life-cycle modeling (45–47%).

Concurrency viewpoint. 66% of the participants model the concurrency views of their software systems in UML. The top-used UML diagrams are (*i*) the class diagram for the concurrency structure modeling (75%) and (*ii*) the component diagram for modeling the mapping between the functional components and the concurrent components (35%).

Development viewpoint. 64% of the participants model the development views of their software systems in UML. The top-used UML diagrams are (*i*) the component and package diagrams for the software module structure modeling (47%), (*ii*) the class and package diagrams for the source-code structure modeling (30–34%), (*iii*) the activity diagram for the software build process modeling (20%), and (*vi*) the activity diagram for the software release process modeling (22%).

Deployment viewpoint. 75% of the participants model the deployment views of their software systems in UML. UML's deployment diagram is by-far the top-used diagram for modeling the physical structure of software systems (71%), and this is followed by UML's component diagram (29%). The top used UML diagram for modeling the mapping between the functional and physical component is the deployment diagram (53%).

Operational viewpoint. Only 29% of the participants model the operational views of their software systems in UML. The top-used UML diagrams are (*i*) the class and deployment diagrams for the installation element modeling (33-39%), (*ii*) the activity and use-case diagrams for the system administration modeling (30–36%), (*iii*) the component diagram for the system configuration modeling (21%), (*vi*) the use-case and class diagrams for the system support modeling (16%), and (*v*) the deployment diagram for the system migration modeling (21%).

6.2. Lessons learned

In our survey, we essentially used our knowledge and experience from our past surveys [14,17]. So, this really helped us in many aspects of the survey design and execution, including the preparation of the survey questions (e.g., multiple-choice questions, question dependencies), determining the potential participants (e.g., using social media effectively, mailing lists, and reaching the practitioners involved in academic papers), and analysing the survey responses (e.g., coding for free-text answers).

We learned many lessons about how the practitioners who model software architectures using UML approach towards a number of architecture viewpoints proposed by Rozanski et al. [19]. The functional and information viewpoints are particularly the top-popular viewpoints. That is, practitioners are highly interested in modeling with UML (*i*) the functional components that compose the systems and their interactions and (*ii*) how those components store, access, and transmit the data. The deployment viewpoint is also quite popular. Indeed, given UML's explicit support for the deployment viewpoint via its deployment diagram and some other diagrams that can also be used (e.g., composite structure diagram and package diagram), this is understandable. Modeling UML-based software architectures from the concurrency and development viewpoints is shown relatively less interest by practitioners. One important reason here could be UML's lack of support for modeling the concurrency and development issues. Indeed, UML does not actually provide any diagram types for modeling software threads and processes and their concurrent interactions. Moreover, from our experience, most practitioners are not used to planning the development issues (e.g., source-code organisation and software build & release processes) during the modeling and design stage, and that is generally omitted until the implementation stage. Rozanski et al.'s operational viewpoint is ignored by most practitioners. This can again be attributed to practitioners' main focus on using UML for the software requirements and design modeling and their ignorance of the operational decisions about the running system in its environment. However, it is highly crucial to plan and analyse the operational concerns, such as administration, system support, configuration, and installation, and take the necessary actions early on before developing and running the software system in its environment. Indeed, after the system development, dealing with such issues may require much more time and effort.

Another lesson learnt is about the UML modeling tools. While many UML tools are available for practitioners' use, the survey results reveal that Enterprise Architect (EA)[5] is by far the top-used UML modeling tool for any viewpoints considered. While EA is essentially a commercial tool that is not available as open-source, it provides several important facilities for practitioners including simulation, model analysis for OCL constraints, roundtrip engineering, collaboration support, and project management. The other popular UML modeling tools, such as Visual Paradigm, MagicDraw, IBM Rational Rhapsody, StarUML, and Modelio, are rarely used by practitioners. Interestingly, the office tools (e.g., Microsoft Office) seem also to be quite popular among practitioners, who use the office tools for drawing their UML-based software architectures. It should however be noted that the office tools cannot process the UML models for, e.g., syntax checking and code generation, as is the case with the UML modeling tools. Therefore, the UML models created with the office tools may simply be used for documentation and communication purposes only.

In the survey, we also observed that practitioners prefer a range of UML diagrams for each model type that Rozanski et al. defined for each of their architecture viewpoints. However, some UML diagrams are preferred relatively more by practitioners. These are the component, class, deployment, and activity diagrams. UML's class diagram is practitioners' top-choice for the following types of software models that are associated with different viewpoints: (*i*) the functional structure, (*ii*) the data structure, (*iii*) the concurrency structure, (*iv*) the software code structure, and (*v*) the system installation & support. UML's deployment diagram is the top choice for the following types of models: (*i*) the physical structure, (*ii*) the mapping between the functional and physical components, and (*iii*) the system migration. UML's activity diagram is the top-choice for the following types of software models: (*i*) the data flow, (*ii*) the soft-

[5] Enterprise Architect web-site: https://www.sparxsystems.com.au/.

ware build and release processes, and (*iii*) the system administration. Lastly, UML's component diagram is the top-choice for the following types of software models: (*i*) the mapping between the functional and concurrent components, (*ii*) the software module structure, and (*iii*) the system configuration. Note however that it is not possible to understand for any model type what the top-preferred UML diagram(s) is actually used for. Indeed, while UML class diagram is practitioners' top-choice for the functional structure modeling, we cannot know whether practitioners use the class diagram for decomposing systems into functional components or modeling the decomposed components' internal structures. Therefore, another lesson to be learnt here could be that we could have asked the practitioners to further indicate what exactly they specify with the chosen UML diagram for the model type in question so as to enhance precision.

6.3. Threats to validity

6.3.1. Internal validity

The internal validity is concerned with the cause-effect relationships that may be influenced by some unknown variables and lead to biased results. So, to minimise any internal threats, we did not consider any responses that come from the participants with no experience in software development and UML modeling. Indeed, such participants may potentially introduce some unknown variables that affect the analysis of the results. As discussed in Section 4.2, the responses received from the academics with no industry background and from the practitioners who never use UML have been omitted in the survey.

Another source of internal threats is to do with the non-random selection of the participants, which may potentially introduce unknown variables. While we used our personal contacts and the pre-determined list of practitioners from the scientific papers on relevant topics, we got most of the participations from the online social platforms (i.e., linkedin groups, mailing lists, google groups, and forums) that essentially mimic the random selection of the participants. Indeed, the potential participants using those social platforms each have got an equal chance of filling in the survey. As discussed in Section 4.3, the survey received the greatest amount of responses from the relevant linkedin groups and the several mailing lists of Eclipse and IEEE. So, this reveals that most of the survey participants have actually been chosen randomly from among the practitioners who have been using the online social platforms.

Moreover, the survey participants represent the diverse profiles of the community, which essentially reduces the risk of getting affected by any unknown variables that may occur when the participants all represent a particular profile only. Indeed, the participants hold several different job positions that involve software development (e.g., architect, developer, managerial positions, systems engineer, etc.) and work on different types of software projects (e.g., web applications, systems software, business applications, mobile applications, etc.) in different industries (e.g., IT & telecommunications, finance & accounting, automative & transportation, etc.).

6.3.2. External validity

The external validity is concerned with to which extent the survey results can be generalized to the entire population of practitioners who are involved in software development. To minimize any threats against the external validity, we intended to target the participants from diverse profiles. So, our survey has been participated by 109 different practitioners from 34 different countries and those participants vary in their work industries, job positions, software projects, and experiences.

6.3.3. Construct validity

The construct validity is essentially concerned with the extent to which the survey results aid in answering the research questions targeted. To minimise any threats, the participants data have been analysed statistically. That is, for each survey question, the participant votes have been counted and stored in the MS Office Excel tool, which enabled to analyse the data statistically and visualise the analysis results via charts.

The mono-operation bias has been minimised as the survey data have been received from the participants of 34 different countries who are involved in various project types and work in several different industries.

To reduce any biases due to the inexact definitions of constructs, we ignore practitioners' usage of the UML diagrams for the viewpoints modeling in general and rather put our main focus on learning the UML diagrams used for different types of models in each viewpoint that can be specified in the system views (e.g., the data flow, data structure, and data life-cycle model types for the information viewpoint). It should however be noted that we did not aim in our survey to understand for each model type what specific concerns practitioners tend to model with the UML diagrams that they choose. Indeed, it is not clear as to whether the practitioners who specify the functional structure model with the UML class diagram use the class diagram particularly for the system's functional decomposition into components or the components' internal structures.

Moreover, the online survey form informs any potential participants about the goal of the survey and get their consent before starting the survey. So, this avoids any participants from filling the survey without clearly understanding what the survey is about. To further minimise any misunderstanding and ambiguities, we supplemented each survey section that consists of the questions of a particular viewpoint with the official web-site link of Rozanski et al.'s book. As discussed in Section 4.1, these links provide precise information about the viewpoints and can avoid any wrong interpretations of the survey questions by the participants.

Lastly, for each viewpoint section in the survey, the participants have been firstly asked how frequently they use that software architecture viewpoint in their UML modeling. If the participant states that they never use, we direct him/her to the next section and prevent from answering the questions about that viewpoint.

7. Conclusion

The survey discussed in this paper attracted 109 participants from 34 different countries, who represent the different profiles in terms of the work industries, job positions, the types of software projects involved, and years of experiences. According to the survey results, most of the practitioners (88%) use UML in modeling their software systems from different architecture viewpoints. Among the considered architecture viewpoints (i.e., functional, information, concurrency, development, deployment, and operational), the top popular viewpoints in which the participants model their systems using UML are the functional and information viewpoints (96–99%). The operational viewpoint is the least popular one, ignored by 61% of the participants in their software modeling with UML. Each software architecture viewpoint has been considered in terms of different types of models that can be specified in that viewpoint. The survey results reveal the UML diagrams that practitioners prefer for each viewpoint model. The functional viewpoint has been addressed with the functional structure model; the information viewpoint has been addressed with the data flow, data structure, and data life-cycle models; the concurrency viewpoint has been addressed with the concurrency structure model and the mapping model between the functional and concurrency components; the development viewpoint has been addressed with the software module structure, software source-code, and build & release models; the deployment viewpoint has been addressed with the physical structure model and the mapping model between the functional and physical components; lastly, the operational viewpoint has been addressed with the system installation, administration, configuration, support, and migration models. Moreover, Enterprise Architect has been observed to be the top used UML modeling tool regardless of the viewpoints considered. Indeed, the rest of the UML modeling tools have been observed to be rarely used. Note also that

some practitioners seem to use the office tools for specifying their software systems from different architecture viewpoints.

The results of this survey study are planned to be validated via the *XIVT* project,[6] which is labeled by the European Union's EUREKA Cluster programme ITEA (Information Technology for European Advancement). To validate the survey results, a set of case-studies will be designed and executed in different industries that the project partners work in, such as Turkcell for telecommunications, Arcelik for consumer electronics, and Bombardier for aviation. The design of the case-studies and the data collection and analysis will be performed systematically in accordance with the well-established guidelines, e.g., [47]. Each case-study will focus on a real problem that is considered to be crucial for the project partner working in a particular industry of interest and can be solved with a software development. A pre-determined set of practitioners working for the project partner will be expected to create software models to design their solutions from different set of viewpoints. We plan to collect data by observing the techniques and tools that the practitioners prefer to use while modeling and asking the practitioners a pre-determined set of questions during and after their modeling activities. After validating the survey results, we will use the results for our works on designing and developing domain-specific modeling languages for different industries. So, we will determine for any particular domains (e.g., automotive, defense, transportation, and telecommunication) (*i*) the set of modeling viewpoints to be supported, (*ii*) the popular types of viewpoint models in which practitioners tend to create models in that domain, and (*iii*) the types of visual notation sets for the viewpoint models that are more likely to be accepted by practitioners in those industries. Moreover, we also plan to determine and analyse all the existing software architecture viewpoint frameworks that offer a set of viewpoints for the architecture descriptions. By doing so, we aim to compare the existing frameworks based on some pre-determined set of requirements including the scope, the supported viewpoints, the viewpoint descriptions (e.g., the concerns addressed, the different types of models supported), and the support for the viewpoint consistencies.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Mert Ozkaya: Conceptualization, Methodology, Data curation, Investigation, Writing - original draft, Writing - review & editing, Visualization. **Ferhat Erata:** Methodology, Investigation, Writing - review & editing.

Acknowledgments

We would like to thank all the participants who contributed to our survey. Moreover, we also would like to express our sincere gratitude to Nick Rozanski and Eoin Woods for their inspiring book on software architecture viewpoints that motivates us for the survey. Also, we would like to thank again Eoin Woods for his highly useful feedback regarding the survey design & execution and his help on disseminating the survey to many practitioners.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.infsof.2020.106275.

[6] XIVT web-site: https://itea3.org/project/xivt.html.

References

[1] D.E. Perry, A.L. Wolf, Foundations for the study of software architecture, SIGSOFT Softw. Eng. Notes 17 (4) (1992) 40–52, doi:10.1145/141874.141884.
[2] D. Garlan, M. Shaw, An introduction to software architecture, Technical Report, Pittsburgh, PA, USA, 1994.
[3] P.C. Clements, D. Garlan, R. Little, R.L. Nord, J.A. Stafford, Documenting software architectures: views and beyond, in: L.A. Clarke, L. Dillon, W.F. Tichy (Eds.), ICSE, IEEE Computer Society, 2003, pp. 740–741.
[4] P. Kruchten, The 4 + 1 view model of architecture, IEEE Softw. 12 (6) (1995) 42–50, doi:10.1109/52.469759.
[5] D. Soni, R.L. Nord, C. Hofmeister, Software architecture in industrial applications, in: Proceedings of the 17th International Conference on Software Engineering, in: ICSE '95, ACM, New York, NY, USA, 1995, pp. 196–207, doi:10.1145/225014.225033.
[6] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, R. Little, Documenting Software Architectures: Views and Beyond, Pearson Education, 2002.
[7] J. Garland, R. Anthony, Large-Scale Software Architecture: A Practical Guide Using UML, first ed., Wiley Publishing, 2002.
[8] M.W. Maier, D. Emery, R. Hilliard, Software architecture: introducing ieee standard 1471, Computer 34 (4) (2001) 107–109, doi:10.1109/2.917550.
[9] M. Ozkaya, The analysis of architectural languages for the needs of practitioners, Softw. Pract. Exp. 48 (5) (2018) 985–1018, doi:10.1002/spe.2561.
[10] Z. DeVito, N. Joubert, F. Palacios, S. Oakley, M. Medina, M. Barrientos, E. Elsen, F. Ham, A. Aiken, K. Duraisamy, E. Darve, J. Alonso, P. Hanrahan, Liszt: a domain specific language for building portable mesh-based PDE solvers, in: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, in: SC '11, ACM, New York, NY, USA, 2011, pp. 9:1–9:12, doi:10.1145/2063384.2063396.
[11] R. Membarth, O. Reiche, F. Hannig, J. Teich, M. Krner, W. Eckert, Hipacc: a domain-specific language and compiler for image processing, IEEE Trans. Parallel Distrib. Syst. 27 (1) (2016) 210–224, doi:10.1109/TPDS.2015.2394802.
[12] C.L. Conway, S.A. Edwards, NDL: a domain-specific language for device drivers, SIGPLAN Not. 39 (7) (2004) 30–36, doi:10.1145/998300.997169.
[13] J.E. Rumbaugh, I. Jacobson, G. Booch, The unified modeling language reference manual, Addison-Wesley-Longman, 1999.
[14] M. Ozkaya, What is software architecture to practitioners: asurvey, in: S. Hammoudi, L.F. Pires, B. Selic, P. Desfray (Eds.), MODELSWARD 2016 - Proceedings of the 4rd International Conference on Model-Driven Engineering and Software Development, Rome, Italy, 19–21 February, 2016., SciTePress, 2016, pp. 677–686, doi:10.5220/0005826006770686.
[15] A. Forward, T.C. Lethbridge, Problems and opportunities for model-centric versus code-centric software development: a survey of software professionals, in: Proceedings of the 2008 International Workshop on Models in Software Engineering, in: MiSE '08, ACM, New York, NY, USA, 2008, pp. 27–32, doi:10.1145/1370731.1370738.
[16] A. Tiso, F. Ricca, M. Torchiano, G. Reggio, F. Tomassetti, Preliminary findings from a survey on the md state of the practice, in: 2011 International Symposium on Empirical Software Engineering and Measurement(ESEM), 00, 2011, pp. 372–375, doi:10.1109/ESEM.2011.51.
[17] M. Ozkaya, Do the informal & formal software modeling notations satisfy practitioners for software architecture modeling? Inf. Softw. Technol. 95 (2018) 15–33, doi:10.1016/j.infsof.2017.10.008.
[18] M. Ozkaya, Analysing UML-based software modelling languages, J. Aeronaut. Space Technol. 11 (2) (2018) 119–134.
[19] N. Rozanski, E. Woods, Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives, Addison-Wesley Professional, 2005.
[20] B. Bauer, J.P. Müller, J. Odell, AgenT UML: a formalism for specifying multiagent software systems, Int. J. Softw. Eng. Knowl. Eng. 11 (3) (2001) 207–230.
[21] T. Lodderstedt, D.A. Basin, J. Doser, SecureUML: a UML-based modeling language for model-driven security, in: J. Jézéquel, H. Hußmann, S. Cook (Eds.), UML 2002 - the Unified Modeling Language, 5th International Conference, vol. 2460, Dresden, Germany, September 30–October 4, 2002, Springer, 2002, pp. 426–441.
[22] J. Jürjens, UMLsec: extending UML for secure systems development, in: J. Jézéquel, H. Hußmann, S. Cook (Eds.), UML 2002 - the Unified Modeling Language, 5th International Conference, vol. 2460, Dresden, Germany, September 30–October 4, 2002, Springer, 2002, pp. 412–425.
[23] Q.Z. Sheng, B. Benatallah, ContextUML: a UML-based modeling language for model-driven development of context-aware web services, in: ICMB, IEEE Computer Society, 2005, pp. 206–212.
[24] B. Selic, Using UML for modeling complex real-time systems, in: LCTES, in: Lecture Notes in Computer Science, 1474, Springer, 1998, pp. 250–260.
[25] R. Martinho, D. Domingos, J. Varajão, FlexUML: a UML profile for flexible process modeling, in: SEKE, Knowledge Systems Institute Graduate School, 2007, pp. 215–220.
[26] SoaML_1_0_1, Service Oriented Architecture Modeling Language (SoaML) Specification, Version 1.0.1, Object Management Group, 2012.
[27] O.M. Group, Software Process Engineering Metamodel Specification, Technical Report, Object Management Group, 2002.
[28] L. Balmelli, An overview of the systems modeling language for products and systems development, J. Object Tech. 6 (6) (2007) 149–177. www.sysml.org
[29] D. Dori, N. Wengrowicz, Y.J. Dori, A comparative study of languages for model-based systems-of-systems engineering (MBSSE), in: 2014 World Automation Congress (WAC), 2014, pp. 790–796, doi:10.1109/WAC.2014.6936160.
[30] P. Andersson, M. Höst, UML and SystemC – A Comparison and Mapping Rules for Automatic Code Generation, Springer Netherlands, Dordrecht, pp. 199–209, doi:10.1007/978-1-4020-8297-9_14.

[31] R. Bendraou, J. Jézéquel, M. Gervais, X. Blanc, A comparison of six UML-based languages for software process modeling, IEEE Trans. Softw. Eng. 36 (5) (2010) 662–675.
[32] L. Brisolara, L. Becker, L. Carro, F. Wagner, C.E. Pereira, R. Reis, Comparing high-level modeling approaches for embedded system design, in: Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005., 2, 2005, pp. 986–989, doi:10.1109/ASPDAC.2005.1466505.
[33] A. Belghiat, E. Kerkouche, A. Chaoui, M. Beldjehem, Mobile agent-based software systems modeling approaches: a comparative study, CIT 24 (2) (2016) 149–163.
[34] M. Petre, UML in practice, in: Proceedings of the 2013 International Conference on Software Engineering, in: ICSE '13, IEEE Press, Piscataway, NJ, USA, 2013, pp. 722–731.
[35] A. Nugroho, M.R. Chaudron, A survey into the rigor of UML use and its perceived impact on quality and productivity, in: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, in: ESEM '08, ACM, New York, NY, USA, 2008, pp. 90–99, doi:10.1145/1414004.1414020.
[36] S. Wrycza, B. Marcinkowski, A light version of UML 2: survey and outcomes, in: Proceedings of the 2007 Computer Science and IT Education Conference, University of Technology Mauritius Press, 2007.
[37] C. Lange, M. Chaudron, J. Muskens, In practice: UML software architecture and design description, IEEE Softw. 23 (2) (2006) 40–46, doi:10.1109/MS.2006.50.
[38] H. Osman, A. van Zadelhoff, D.R. Stikkolorum, M.R.V. Chaudron, UML class diagram simplification: what is in the developer's mind? in: Proceedings of the Second Edition of the International Workshop on Experiences and Empirical Studies in Software Modelling, in: EESSMod '12, ACM, New York, NY, USA, 2012, pp. 5:1–5:6, doi:10.1145/2424563.2424570.
[39] A.M. Fernndez-Sez, D. Caivano, M. Genero, M.R.V. Chaudron, On the use of UML documentation in software maintenance: results from a survey in industry, in: 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS), 2015, pp. 292–301, doi:10.1109/MODELS.2015.7338260.
[40] B. Dobing, J. Parsons, Current practices in the use of UML, in: Proceedings of the 24th International Conference on Perspectives in Conceptual Modeling, in: ER'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 2–11, doi:10.1007/11568346_2.
[41] M. Grossman, J.E. Aronson, R.V. McCarthy, Does UML make the grade? Insights from the software development community, Inf. Softw. Technol. 47 (6) (2005) 383–397, doi:10.1016/j.infsof.2004.09.005.
[42] G. Reggio, M. Leotta, F. Ricca, Who knows/uses what of the UML: a personal opinion survey, in: J. Dingel, W. Schulte, I. Ramos, S. Abrahão, E. Insfran (Eds.), Model-Driven Engineering Languages and Systems, Springer International Publishing, Cham, 2014, pp. 149–165.
[43] L.T.W. Agner, I.W. Soares, P.C. Stadzisz, J.M. SimãO, A Brazilian survey on UML and model-driven practices for embedded software development, J. Syst. Softw. 86 (4) (2013) 997–1005, doi:10.1016/j.jss.2012.11.023.
[44] R. Popping, Analyzing open-ended questions by means of text analysis procedures, Bull. Sociol. Methodol. 128 (1) (2015) 23–39, doi:10.1177/0759106315597389.
[45] T. Punter, M. Ciolkowski, B.G. Freimut, I. John, Conducting on-line surveys in software engineering, in: 2003 International Symposium on Empirical Software Engineering (ISESE 2003), 30 September–1 October 2003. Rome, Italy, IEEE Computer Society, 2003, pp. 80–88, doi:10.1109/ISESE.2003.1237967.
[46] R.L. Scheaffer, W. Mendenhall, L. Ott, Elementary Survey Sampling, Duxbury Press, North Scituate, MA, 1986.
[47] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2) (2009) 131–164, doi:10.1007/s10664-008-9102-8.