Article

An Analysis of the Features of Requirements Engineering Tools

Mert Ozkaya [1,*] , Geylani Kardas [2] and Mehmet Alp Kose [3]

1   Computer Engineering Department, Yeditepe University, Istanbul 34755, Turkey
2   International Computer Institute, Ege University, Izmir 35100, Turkey; geylani.kardas@ege.edu.tr
3   Independent Researcher, Istanbul 34710, Turkey; malpkose@gmail.com
*   Correspondence: mozkaya@cse.yeditepe.edu.tr

Abstract: Many requirements engineering tools have been developed for gathering, documenting, and tracing requirements that can even be further processed for such purposes as analysis and transformation. In this study, we analysed 56 different requirements engineering tools for a comprehensive set of features that are categorised into multiple viewpoints (i.e., project management, specification, collaboration, customisation, interoperability, methodology, and user-support). The analysis results led to many interesting findings. Some of them are as follows: (i) the project planning and execution activities are rarely supported, (ii) multi-user access and versioning are highly supported, (iii) the most popular specification technique is natural languages, while precise specification via modeling languages is rarely supported, (iv) requirements analysis is rarely supported, (v) requirements transformation is considered for generating documents only, (vi) tool customisation via the tool integration and API support is highly popular, while customising the notation set is rarely supported, (vii) exchanging requirements is popular in such standards as ReqIF and Excel/CSV, while no single standard is accepted by all the tools, (viii) agile development is very common, while other methodologies (e.g., MDE and SPLE) are rarely supported, and (ix) user-guides, telephone, e-mail, and videos are the most preferred methods for user-support. The analysis results will be useful for different stakeholders including practitioners, tool vendors, and researchers.

Keywords: requirements engineering; tools; survey; viewpoints

1. Introduction

A requirement is considered as a statement that needs to be agreed to by all the stakeholders and contributes to solving customers' problems [1–3]. Each requirement describes either what the system to be developed is expected to perform or any constraints on the system design and development (e.g., quality and platform constraints). Requirements engineering has been proposed as a branch of software engineering that promotes the application of well-known techniques, practices and methods for eliciting, documenting, and analysing requirements [4–7]. With requirements engineering, the goal is to maximise the quality of requirements that will affect the subsequent stages of software development. This involves such activities as understanding the customers' needs completely and correctly, specifying requirements precisely in a verifiable way to avoid any issues (e.g., incomplete and inconsistent requirements), and associating requirements with other artefacts (e.g., test-cases and system architectural components). It should be noted that failing to apply the activities of requirements engineering may lead to wrong systems being developed. Indeed, many software projects fail because of the ill-defined application of the requirements engineering practices [8–10].

To perform the requirements engineering activities effectively, several tools are available on the market through which the requirements for any systems can easily be specified and managed collaboratively throughout the project's lifecycle. Requirements engineering tools offer diverse features, including project management facilities, requirements traceability, automated analysis, document generation, test-scenario generation, multi-user

collaboration, and importing/exporting using the ReqIF standard [11], an API support for customising the tool.

While tens of different requirements engineering tools are available, the literature lacks any resources that can be used for determining the existing requirements engineering tools, understanding the tools' support for a number of important features, and comparing the tools with each other. As discussed in Section 5, the existing studies that attempt to compare the requirements engineering tools either ignore many of the existing tools, focus on a small set of features, or consider just the tools in particular domains (e.g., tools for natural language processing).

Therefore, in this paper, we aim to discuss the results of our literature review on the analysis of 56 different requirements management tools with the considerations of a comprehensive set of tool features. We categorised the features that we considered into multiple viewpoints: project management, specification, collaboration, customisation, interoperability, methodology, and user-support. The project management viewpoint is concerned with the support for initiating, planning and executing the projects. The specification viewpoint is concerned with the technique(s) supported for specifying requirements (e.g., natural languages and modeling languages), the support for analysing requirements automatically, and the support for transforming requirements into useful artefacts (e.g., code and test scenarios). The customisation viewpoint is concerned with customising and extending the requirements engineering tools for the specific user needs. The interoperability viewpoint is concerned with the support for importing/exporting the requirement specifications between different tools. The methodology viewpoint is concerned with the support for any well-known software development methodologies that can facilitate the requirements engineering activities such as agile development, product-line engineering, and model-based engineering. Lastly, the user-support viewpoint is concerned with supporting users in their tool usages through which the learning curve can be minimised.

The analysis results discussed in our paper are believed to provide invaluable guidance for the stakeholders who are involved in the requirements engineering. Indeed, while the existing literature includes some review studies on the requirements engineering tools, none of those studies consider as many tools and focus on as many tool features as we do in our review study here. Also, our review is further unique with its consideration of several interesting viewpoints, such as project management, interoperability, different methodologies of software development, and user-support, each of which is represented with a cohesive set of features. We strongly believe that stakeholders can gain some interesting insights from the review results about the requirements engineering and the tool support. Practitioners in the industry may use the analysis results to identify many of the existing requirements engineering tools, observe their weak and strong points in terms of the practical set of features, compare different tools, and choose the ones that best meet their needs. Tool vendors can use the results to understand to what extent their tools satisfy different principal concerns for requirements engineering and any gaps that can be improved. Researchers in the field of requirements engineering can conduct further empirical studies on such concerns as understanding practitioners' perspectives towards the requirements engineering tools, analysing a subset of tools for a particular viewpoint, comparing different types of tools for the requirements engineering, etc.

In the rest of the paper, we firstly give the research methodology that we applied in our study. Then, we discuss the analysis results of the 56 different tools for a set of viewpoints. Then, we give the summary of findings and the lessons learned, which are followed by the discussions of any threats to the validity of the analysis results.

2. Research Methodology

2.1. Review Protocol

We performed our literature review using the PRISMA guidelines [12] and applied the checklist provided by PRISMA for systematic review[1].

Conforming to the PRISMA guideline's introduction section, we previously discussed the motivation and objectives of our review study (see Section 1). In this section, we first describe how we conducted the review according to the PRISMA method's definitions. In Section 2.2, we give the search strategy. In Section 2.3, we give the eligibility criteria. In Section 2.4, we discuss the very first stage of the data collection process according to PRISMA specifications and define the set of features for which we collect data about the tools. Then, in Section 2.5, we explain how we keep the data sources for each tool and the protocol that we follow for analysing the data sources. We also discuss in Section 2.5 the risk of any biases due to the different reviewers contributing to the data collection process. Lastly, in Section 2.6, we discuss how we synthesise the collected tool data for reaching interesting findings.

Later, in Sections 3 and 4, we discuss the results that we reached through our syntheses of the tool data and indicate the implications of our results and any lessons learned with our study, respectively, again by following the PRISMA guideline.

2.2. Search Strategy

In this section, we discuss the search strategy that we followed for reaching as many requirements engineering tools as possible. We discuss the search scope (i.e., the search period and search platform), search method (i.e., manual/automated search), and search query (i.e., the strings used for searching tools) subsequently.

2.2.1. Search Scope

In this review study, we aimed to search for the requirements engineering tools that can be accessible via the popular search engine *Google*. We performed our tool search via *Google* between March 2023 and June 2023.

2.2.2. Search Method

We used the *Google* search engine manually so as to reach as many tools as we could. That is, we firstly used *Google* with the general search query given in the next section and scanned through all the pages that *Google* returned manually. We performed the same for each specific search query given in the next section. Moreover, we performed snowballing in our search as we scanned through the web pages that *Google* returned and gave a list of the requirements engineering tools. Table 1 gives the list of those web pages.

Table 1. The webpages that present a list of requirements engineering tools.

| Name	URL	Access Date
List of requirements engineering tools	https://en.wikipedia.org/wiki/List_of_requirements_engineering_tools	9 April 2023
7 Requirements engineering tools to make your life easy	https://www.zumvie.com/7-requirements-engineering-tools-to-make-your-life-easy/	9 April 2023
10 Best Requirements Management Tools & Software of 2023	https://thedigitalprojectmanager.com/tools/requirements-management-tools/	9 April 2023
Top 20+ Best Requirements Management Tools	https://www.softwaretestinghelp.com/requirements-management-tools/	9 April 2023
13 BEST Requirements Management Tools & Software (2023)	https://www.guru99.com/requirement-management-tools.html	9 April 2023
Software Requirements Engineering Tools	https://ecomputernotes.com/software-engineering/softwarerequirementsengineeringtools	9 April 2023
Top Requirements Management Tools List	https://blog.testlodge.com/requirements-management-tools-list/	9 April 2023

2.2.3. Search String

To find the requirements engineering tools, we initially used following general search query: ("requirements") AND ("engineering" OR "management" OR "specification" OR

"modeling") AND ("software" OR "tool" OR "technology" OR "platform" OR "language"). Besides, we also used several specific queries that are related to the features for which the tools are to be analysed. These include "requirements engineering tools/platforms", "requirements management tools/platforms", "requirement modeling tools/platforms", "requirement tools/platforms", "requirements modeler", "SDLC tools", "modeling tools/platforms", "software specification tools/platforms", "agile requirements management", "agile requirements tools", "model-driven requirements management", "model-driven engineering tools", "product line requirements management", and "product line engineering tools".

2.3. Eligibility Criteria

Figure 1 shows the PRISMA flow diagram[2] that depicts the eligibility criteria for the requirements engineering tools that we identified using the search strategy discussed above. It is worth indicating that among four different variations of PRISMA flow diagrams, we preferred and exactly followed the one here, which is specific for the new systematic reviews including searches of databases and registers only. Therefore, we managed to obtain 200 different modeling tools in total and then eliminated 10 tools as they were essentially the duplicates of some other tools. Among the remaining 190 tools, we eliminated 25 tools as we were not able to reach their websites. We further excluded any tools that did not provide direct support for managing software requirements (e.g., the modeling tools that support UML modeling). We also excluded any tools for which we were not able to reach any download link for a trial version, available resources (e.g., user manual, white papers, case-studies, publications, etc.), and an online editor. Thus, we ended up with a list of 56 different tools, given in Table 2, which we can analyse for the features of interest.
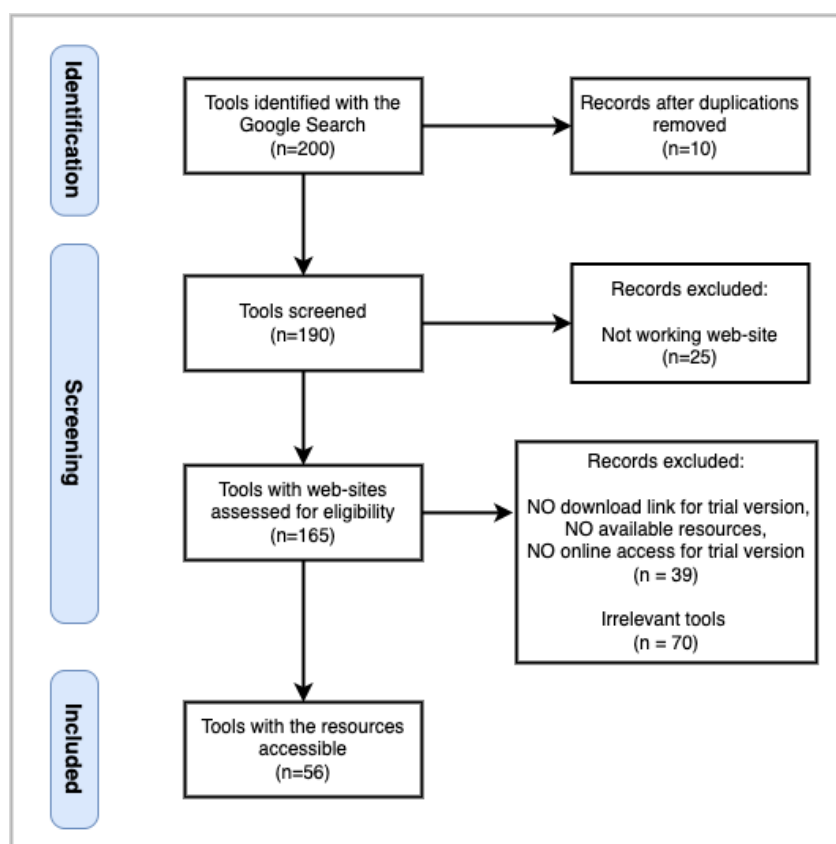
Figure 1. The PRISMA flowchart diagram for the tool screening and reviewing process.

Table 2. The requirements engineering tools.

Tool	Website	Supported Platforms	Open-Source	Year
Accompa PM	www.web.accompa.com	Web	No	2009
acunote	www.acunote.com/	Web	No	2006
Agile Requirements Designer	www.broadcom.com/products/software/continuous-testing/agile-requirements-designer	Web and On-premise	No	2020
agosense.fidelia	www.agosense.com/	Web and On-premise	No	2009
Aha!	www.aha.io/	Web	No	2013
Aligned Elements	www.aligned.ch/features/requirement-management	Web and On-premise	No	2006
Quality Center—Dimensions RM	www.microfocus.com/en-us/products/dimensions-rm/overview	Web	No	2020
Auros IQ	www.aurosks.com/	Web	No	2010
Axosoft	www.axosoft.com/	Web and On-premise	No	2014
Azure DevOps	www.azure.microsoft.com/	Web and On-premise	No	2005
Balsamiq Wireframes	www.balsamiq.com/	Web and On-premise	Yes	2008
Business Optix	www.businessoptix.com/	Web	No	2010
Cameo Systems Modeler	www.3ds.com/products-services/catia/products/no-magic/cameo-systems-modeler/	Web and On-premise	No	2014
Capella	www.eclipse.org/capella/	On-premise	Yes	2023
CaseComplete	www.casecomplete.com/	Web	No	2012
ClickUp	www.clickup.com/	Web and On-premise	No	2017
CodeBeamer ALM	www.codebeamer.com/	Web and On-premise	No	2002
Cradle	www.threesl.com/cradle/	Web and On-premise	No	2015
Doc Sheets	www.docsheets.com	Web and On-premise	No	2000
Eclipse Papyrus	www.eclipse.org/papyrus	On-premise	Yes	2019
Enterprise Architect	www.sparxsystems.com/	On-premise	No	2000
Helix RM	www.perforce.com/products/helix-alm	Web and On-premise	No	2016
innoslate	www.innoslate.com/	Web and On-premise	No	2013
Innovator for Business Analysts	www.innovator.de/en/	On-premise	No	2021
in-STEP BLUE	www.microtool.de/en/products/in-step-blue/	Web and On-premise	No	2014
iRise	www.irise.com/	Web and On-premise	No	2016
Jama Connect	www.go.jamasoftware.com/	Web and On-premise	No	2007
Kovair ALM	www.kovair.com	Web	No	2006
MagicDraw	www.3ds.com/products-services/catia/products/no-magic/magicdraw/	On-premise	No	1998
Matrix ALM/QMS	www.matrixreq.com/en/product	Web	No	2013
Modelio Analyst	www.modeliosoft.com/en/modules/analyst.html	Web and On-premise	No	2009
OpenProject	www.openproject.org/	Web and On-premise	No	2012
Orcanos	www.orcanos.com	Web and On-premise	No	2004
PivotalTracker	www.pivotaltracker.com/	Web	No	2008
Polarion Requirements	www.polarion.plm.automation.siemens.com/products/polarion-requirements	Web	No	2010
Psoda	www.psoda.com/	Web	No	2008
Rational DOORS	www.ibm.com/docs/en/ermd/	Web and On-premise	No	1993
Rational Rhapsody	www.ibm.com/products/uml-tools	On-premise	No	2011
ReqEdit	www.reqteam.com/	On-premise	No	2014
ReQtest	www.reqtest.com/	Web	No	2009

Table 2. *Cont.*

Tool	Website	Supported Platforms	Open-Source	Year
ReqView	www.reqview.com/	Web and On-premise	No	2015
RMsis	www.marketplace.atlassian.com/apps/30899/rmsis-requirements-management-for-jira	Web	No	2010
ReqChecker	https://reqchecker.eu/	On-premise	No	2016
RMTrack	www.rmtrack.com/	Web	No	2002
Scrumwise	www.scrumwise.com/	Web	No	2009
SpiraTeam	www.inflectra.com/SpiraTeam/	Web and On-premise	No	2006
StoriesOnBoard	https://storiesonboard.com/	Web	No	2015
SwiftKanban	www.nimblework.com/	Web and On-premise	No	2011
Targetprocess	www.targetprocess.com/	Web and On-premise	No	2006
TopTeam	www.topteamrequirements.com	Web and On-premise	No	1995
Tuleap Enterprise	www.tuleap.org	Web and On-premise	No	2011
Valispace	www.docs.valispace.com/	Web and On-premise	No	2016
Visual Paradigm	www.visual-paradigm.com	Web and On-premise	No	2002
Visure Requirements	www.visuresolutions.com/	Web	No	2007
Yodiz	www.yodiz.com	Web	No	2010
Xebrio	www.xebrio.com	Web and On-premise	No	2018

2.4. Identifying the Tool Features

Our data collection process started with identifying the tool features for which the tool data were to be collected.

To determine the features for which the requirements engineering tools shown in Table 2 can be examined, we used the results of our past relevant studies in the field. Indeed, in [13], we surveyed among 84 practitioners from diverse industries to understand the practitioners' perspectives on the requirements engineering. The survey results helped us understand the practitioners' motivations for the requirements engineering, the techniques and technologies used for different activities of requirements engineering, practitioners' experiences with customer involvement, and their challenges. In another study [14], we analysed 58 UML modeling tools for a number of tool features that are considered important for the practical use of the UML tools, which are modeling viewpoints, analysis, transformation and export, collaboration, tool integration, scripting, project management, and user-support. Also in [15], we analysed 124 different modeling languages for a comprehensive set of features, which are multiple viewpoints, non-functional properties, notation sets, formal semantics, extensibility, programming framework, automated analysis, large-view management, collaboration, versioning, and user-support. Thus, we essentially went through *(i)* a high number of different feature candidates that we addressed in our previous related studies and *(ii)* the survey results on the practitioners' perspectives. Finally, we filtered our existing feature set with regard to the relevance to the requirements engineering and ended up with a subset of features that could be used for better understanding and comparing the existing requirements engineering tools and reaching important findings. To facilitate the understandability, we categorised the features that we focused on into multiple viewpoints, which are project management, collaboration, requirements specification, customisation, interoperability, methodology, and user-support. Within this context, it is worth indicating that it could be an option to benefit from only a set of standard features existing for the requirement engineering tools (as successfully utilized in various previous studies such as [16]). However, in our study, we prefer considering both these existing standard features that vary from requirement engineering definition to requirement management as well as defining new ones to cover other aspects, such as applied software engineering

methodology and tool customization, which we believe have significant effects on the planned tool evaluation.

2.4.1. Project Management

Project management is considered a highly important factor for the effective management of requirements, and the proper support for managing projects needs to be considered for requirements engineering [17–19]. With the project management viewpoint, we focus on three project management stages, which are project initiation, project planning, and project execution [20]. Project initiation can be considered as the first stage of project management and is concerned with the activities that are performed once the project is approved, including defining the problem, solution, and scope in detail, assigning project manager and creating teams, and organising a physical area. Project initiation is followed by the project planning, which is concerned with drawing a Gantt chart for planning the tasks and their dependencies, and other issues such as resource planing, budget planning, staff planning, and risk planning. Project execution is concerned with the stage where the project plans are realised and any progresses are reported.

2.4.2. Collaboration

With the collaboration viewpoint, we focus on the capability of multiple users using the same tool with different access rights collaboratively. We consider the collaboration support in terms of multi-user access, user role definitions, user-access right definitions, and requirements versioning. The multi-user access support enables multiple users to access the same project and manage the requirements together at the same time. The support for the user roles enables the definition of the roles of the users such as customer, analyst, system engineer, developer, manager, etc. The support for the user-access right enables the restriction of user access in such ways as read-only access, edit access, authorisation, etc. The support for versioning enables different versions of the requirements to be kept in a repository where different versions can be accessed, compared, and even deleted by the users.

2.4.3. Requirement Specifications

With the requirements specification viewpoint, we are concerned with the specification of requirements, the analysis of requirements specifications, and the transformation of requirements specifications into some useful artefacts such as code, test scenarios, and documentation.

For the requirement specifications, we consider three important techniques suggested by Taylor et al. [21]—natural languages, boxes-and-lines, and modeling languages. Natural languages and boxes-and-lines essentially represent the informal techniques for specifying requirements and promote any stakeholders without technical knowledge to be involved in the requirements specification process. While natural languages may be used via some structured templates to enhance the precision and facilitate the requirements analysis, the pure (i.e., unstructured) use of natural languages (or boxes-and-lines) can indeed lead to ambiguous specifications that are interpreted differently by different stakeholders due to the lack of precise definitions (i.e., syntax and semantics).

Another method for specifying requirements is modeling languages. Modeling languages can be supported with precise (and sometimes formal) definitions (i.e., syntax and semantics) and thus enable the nonambiguous communications of requirements among stakeholders. Also, with some modeling tool support, it can be easier to analyse the requirement specifications and even generate executable code and test scenarios. Modeling languages can be general-purpose (e.g., UML [22], SysML [23], and BPMN [24]) or domain-specific (e.g., AADL for embedded systems [25]). Modeling languages are supported by several tools (e.g., UML tools [14] and AADL's OSATE[3]), which can aid in specifying requirements, analysing the specifications, and transforming them into code.

Concerning the requirements analysis, we consider the tool support for a predefined set of properties for which the requirement specifications can be analysed: *(i)* defining new (i.e., user-defined) properties for which the requirements specifications can be analysed, *(ii)* simulating (i.e., executing) the requirement specifications, and *(iii)* completeness and consistency checks [26]. The requirement consistency here can be used to check if two requirements conflict with each other (e.g., implying the same functionality), while the requirement completeness can be used to check if all the requirements are specified and each requirement is specified with no missing information.

Concerning the requirements transformation, we consider the support for generating *(i)* skeleton code from requirements, *(ii)* test scenarios from requirements, and *(iii)* documents in different formats (e.g., HTML, PDF, Excel, Word, etc.).

2.4.4. Customisation

Customisation of software systems encourages the changing of the software systems with the least effort possible (e.g., adding new components or replacing/changing an existing component) [27]. Thus, with the customisation viewpoint, we are concerned with learning to what extent the requirements engineering tools support any extensions for the users. By doing so, the requirements engineering tools can be used for some domain-specific problems such as the the analysis of requirements for particular properties and the specification of requirements using a domain-specific notation set.

Inspired from Lago et al.'s approach towards extensibility and customisation [28], we consider customisation in terms of the user-defined modeling viewpoints, integration with external tools, API support, and DSL support. With the user-defined modeling viewpoint support, developers can use the modeling elements supported by the requirements engineering tool and define a new viewpoint in terms of the rules and constraints specific to their concern. Then, users can use their viewpoint definition to specify domain-specific requirements [29]. With the integration support, users can integrate the requirements engineering tools with some external tools that exhibit different capabilities such as test automation, design, simulation, verification, project management, versioning, and repository management. With the API support, users can extend the requirements engineering tools by developing new functionalities. With the DSL support, users can develop their own domain-specific modeling languages for specifying requirement models in terms of domain-specific concepts, relationships, and their symbols. Indeed, some tools can enable the use of the UML profiling mechanism for extending the UML for domain-specific purposes [30].

2.4.5. Interoperability

Interoperability is considered as one of the most important quality attributes of software engineering, and it promotes different tools to work together and exchange information [31,32]. With the interoperability viewpoint, we are concerned with the requirements engineering tools' capabilities for exchanging the requirements specifications with other tools. That is, we consider the requirements engineering tools' support for importing/exporting requirement specifications using *(i)* some commonly used data exchange formats (e.g., MS Word, MS Excel, CSV, XMI, and XML) and *(ii)* some well-accepted standards (e.g., the ReqIF).

2.4.6. Methodology

Requirements engineering can be facilitated with some software design and development methodologies. Indeed, methodologies such as agile development [33] and model-driven development [34] can aid in facilitating the requirements engineering activities (e.g., specifications, analysis, and transformation) and the application of software engineering principles to better manage requirements (e.g., software re-use, customer involvement, and quality properties). Also, some methodologies, such as product-line engineering [35], are considered with the effective management of requirements.

In our literature review, we focus on three methodologies—i.e., agile, model-driven engineering, and product-line engineering, which are related to the requirements engineering and are highly popular in industry and academia [36–38]. The agile methodology promotes the requirements engineering activities to be performed with the involvements of customers in an iterative and incremental way so as to manage any change requests effectively [33,39]. Agile development is supported with such techniques as Scrum, Kanban, and lean management, through which the agile principles can be applied effectively. Model-driven engineering (MDE) promotes specifying abstract models (e.g., requirements models) for different perspectives of system development (e.g., structure, interaction, and behaviour) that can further be processed via tool support (e.g., model simulation, code generation, document generation, and versioning). Software product-line engineering (SPLE) promotes the development of software products that have commonalities (e.g., the same functionalities) and variations. SPLE essentially considers the specifications of the commonalities and variabilities of the products (or parts) composing systems and their requirements. As indicated in [40], requirements engineering has huge importance for SPLE as one needs to determine common and variable set of requirements among the products and perform their effective management.

In this review study, we aim to understand basically if the tools support either of these three methodologies or not. We strongly believe that knowing about the tools that support different methodologies can give useful insights for practitioners and even trigger any future empirical research on requirements engineering tools. Note that for simplicity reasons, we only focus on these three methodologies exclusively. Also, considering our scope on analysing requirements engineering tools in general, we do not present any detailed analysis of the tools for their methodology support.

2.4.7. User-Support

With the user-support viewpoint, we are concerned with the support that the requirements engineering tools provide via their websites so as to reduce the time taken to learn and use the tools. We consider different methods for the user-support, including telephone, e-mail, forum, live-chat, help-desk, user-guide, blog, white papers, mailing lists, case-studies, videos, training, and coaching/consulting.

2.5. Collecting Data

After identifying the tools features, we focused on collecting the tool data for the features of interest.

Firstly, we prepared the data collection repository from which the tool resources can be accessed easily. Thus, we created a repository in a shared drive that consists of a separate folder for each tool. The folder for a tool includes *(i)* a file that has the website access information for reaching the tool materials, *(ii)* the tool installation guide documents, and *(iii)* any publications, user manuals, white papers, case-studies, etc. The repository also includes an MS Excel file for storing the tool data with regard to the viewpoints introduced in Section 2.4.

To collect data for each requirements engineering tool, we used the following data collection protocol. We initially installed and used the tool to see which viewpoint features were supported and how they were supported. If we were unable to reach the tool or could not figure out the support for a particular feature using the tool, we used the resource documents for that tool (e.g., website, user-guides, tutorials, white papers, blogs, and videos). That is, we firstly went through the tool website for any useful information for a particular feature. If we could not find any information available on the tool website, we chose to consult the tool's supporting documents. If we could not access the information in any documents either, we tried sending an e-mail to the tool vendors as a last resort (assuming that contact information was available on the website). Note that in the case when we were not able to reach any useful information using the existing materials of a tool, we considered that the associated tool did not provide support for that feature.

Otherwise, we clearly indicated that the tool supported the viewpoint feature and further gave some precise data about the support provided. In the Excel file created (see the previous paragraph), we kept a record of each tool's data. The Excel file included a separate sheet for each viewpoint, where 56 different rows (i.e., one for each tool) and as many columns as the number of viewpoint features were defined. Thus, for each tool, we added a record to each sheet of the Excel file.

To minimise any risk of bias on the collected tool data, the data collection protocol discussed above was performed by the three authors independently. Firstly, each author created their own Excel file which included the data that the author collected for the tools individually. Then, a session was held together with all the authors to merge the collected data by the authors into a single Excel file. In this session, the goal was essentially to compare for each tool the data that were collected by the three authors and check if there were any discrepancies. Whenever a discrepancy was detected for a tool, we added the name of the tool and the feature in question (e.g., the import–export support) into a separate list of discrepancies. After analysing the authors' dataset collected for each tool, we ended up with a complete list of discrepancies. Then, we conducted another session together and considered each discrepancy separately. For each discrepancy, we used the list to identify the name of the tool and the feature in question and discussed what could have caused the inconsistency (e.g., the lack of resources, misinterpretations, etc.). Upon reaching a consensus on the cause, we made a decision on the actions that needed to be taken so as to resolve the issue (e.g., re-examining the existing materials, searching for a new material, contacting the tool vendors, etc.). We also kept the list of discrepancies updated with the actions decided for each discrepancy. One of the authors was chosen to handle the discrepancy list and carry out the agreed actions for each discrepancy in the list. The author made the necessary revisions, documented the changes, and obtained the approval of the other two authors in another short session. Lastly, the merged Excel file was updated and finalised accordingly[4].

2.6. Synthesising Data

After collecting and analysing the data, we focused on synthesising the data so as to obtain some important findings about the requirements engineering tools' support for a comprehensive set of features. That is, we created a table for each viewpoint (i.e., the related set of features) using the MS Excel office tool; thus, the data for the tools that support the features of the viewpoint in question are displayed in a tabular format. Also, we created some pie charts to indicate some important findings about the tools' support for the viewpoint features. Note that during the data analysis process, we had also needed to recollect the data about some tools as we determined some ambiguities and inconsistencies.

3. Results

In this section, we discuss the analysis results of the 56 different requirements engineering tools given in Table 2. We consider each viewpoint introduced in Section 2.4 and discuss the tool support for the viewpoint features.

3.1. Project Management

As shown in Figure 2, 46% of the requirements engineering tools support all the features of project management that are considered (i.e., project initiation, project planning, and project execution). A total of 38% of the tools just support the initiation phase of the project, which includes such activities as creating projects and assigning members to the project. However, those tools do not support planning the project tasks with, e.g., a Gantt chart, and the execution stage. A total of 16% of the tools do not provide built-in support for project management. Note that those tools with no built-in support may provide integration with project management tools such as Jira[5].

Balsamiq Wireframes
Business Optix
Capella
Papyrus
Helix RM
Rational DOORS
Rational Rhapsody
Innovator for Business Analysts
ReqChecker

agosense.fidelia
Aha!
Quality Center - Dimensions RM
Auros IQ
Axosoft
Azure DevOps
Cameo Systems Modeler
codeBeamer ALM
Enterprise Architect
innoslate
in-STEP BLUE
Jama Connect
Kovair ALM
MagicDraw
OpenProject
Polarion Requirements
Psoda
SwiftKanban
Targetprocess
Tuleap Enterprise
Valispace
Visual Paradigm
Yodiz
ClickUp
Doc Sheets
Xebrio

16%

■ No support

46%

■ Project initiation only

38%

■ Project initiation, planning, and execution

acunote
Aligned Elements
CaseComplete
Cradle
iRise
Matrix ALM/QMS
Modelio Analyst
PivotalTracker
ReqEdit
ReQtest
ReqView
RMsis
RMTrak
Scrumwise
SpiraTeam
Visure Requirements
Accompa PM
CA Agile Requirements Designer
Orcanos
TopTeam
StoriesOnBoard

Figure 2. The support for the project management.

3.2. Collaboration

All the requirements engineering tools enable the multiple users to work together on the same project. The only exception here is ReqChecker. A total of 80% of the tools further support assigning roles to the users and giving them appropriate access rights. The tools that ignore either the specifications of user role or access rights are given in Table 3.

Table 3. The requirements engineering tools that do not support either user roles or user-access rights.

Requirements Engineering Tools	Multi-User Access	User Roles	User-Access Rights
acunote	Yes	No	Yes
agosense.fidelia	Yes	No	No
Balsamiq Wireframes	Yes	No	No
Business Optix	Yes	No	No
CaseComplete	Yes	No	No
Eclipse (IDE) Papyrus	Yes	No	Yes
Psoda	Yes	Yes	No
ReqView	Yes	No	No
Valispace	Yes	No	Yes
Orcanos	Yes	No	No
Xebrio	Yes	No	Yes

Almost all the requirements engineering tools (93%) support versioning—ScrumWise, RMTrack, and Psoda are the only exceptions. Table 4 shows the requirements engineering tools' versioning support in terms of the support for built-in central versioning system and the integration with external versioning systems (i.e., GIT, SVN, and Mercurial). Thus, apparently, 74% of the tools supporting versioning provide built-in versioning systems for versioning requirements. The support for the integration with external versioning systems

is quite rare (3–25%)—GIT is supported by 25% of the tools, SVN is supported by 20% of the tools, and Mercurial is supported by just 3%. Acunote is the tool that supports all external versioning systems that are considered—GIT, SVN, and Mercurial. Polarion Requirements is the tool that provides an internal versioning system and supports the integration with GIT and SVN at the same time.

Table 4. The requirements engineering tools that support versioning.

Requirements Engineering Tools	Built-in Versioning	GIT	SVN	Mercurial
acunote	No	Yes	Yes	Yes
agosense.fidelia	No	Yes	No	No
Aligned Elements	Yes	No	No	No
Auros IQ	Yes	No	No	No
Axosoft	No	Yes	No	No
Azure DevOps	No	Yes	No	No
Balsamiq Wireframes	Yes	No	No	No
Business Optix	Yes	No	No	No
Cameo Systems Modeler	Yes	No	No	No
Capella	No	Yes	Yes	No
CaseComplete	No	No	Yes	No
codeBeamer ALM	No	Yes	Yes	Yes
Cradle	Yes	No	No	No
Eclipse (IDE) Papyrus	Yes	Yes	No	No
Enterprise Architect	Yes	No	Yes	No
Helix RM	Yes	No	No	No
innoslate	Yes	No	No	No
Innovator for Business Analysts	Yes	No	No	No
in-STEP BLUE	Yes	No	No	No
iRise	Yes	No	No	No
Jama Connect	Yes	No	No	No
Kovair ALM	Yes	No	No	No
MagicDraw	Yes	No	No	No
Matrix ALM/QMS	Yes	No	No	No
Modelio Analyst	Yes	No	No	No
OpenProject	Yes	No	No	No
PivotalTracker	Yes	No	No	No
Polarion Requirements	Yes	Yes	Yes	No
Quality Center—Dimensions RM	Yes	No	No	No
Rational DOORS	Yes	No	No	No
Rational Rhapsody	Yes	No	No	No
ReqChecker	Yes	No	Yes	No
ReqEdit	Yes	No	No	No
ReQtest	Yes	No	No	No
ReqView	Yes	No	Yes	No
RMsis	Yes	No	No	No
SpiraTeam	Yes	No	Yes	No

Table 4. *Cont.*

Requirements Engineering Tools	Built-in Versioning	GIT	SVN	Mercurial
StoriesOnBoard	No	Yes	No	No
SwiftKanban	No	Yes	No	No
Targetprocess	No	Yes	No	No
Tuleap Enterprise	No	Yes	Yes	No
Valispace	Yes	No	No	No
Visual Paradigm	Yes	No	No	No
Visure Requirements	Yes	No	No	No
Yodiz	No	Yes	Yes	No
Accompa	Yes	No	No	No
CA Agile Requirements Designer	Yes	No	No	No
ClickUp	No	Yes	No	No
Doc Sheets	Yes	No	No	No
Orcanos	Yes	No	No	No
TopTeam	Yes	No	No	No
Xebrio	Yes	No	No	No

3.3. Requirements Specification

As Figure 3 shows, most of the languages (73%) support the informal specifications of requirements using natural languages (e.g., English sentences). A few of those tools (Dimensions RM, Auros IQ, Balsamiq Wireframes, Helix RM, and Jama Connect) supplement the natural languages with simple boxes-and-lines for the requirement specifications.
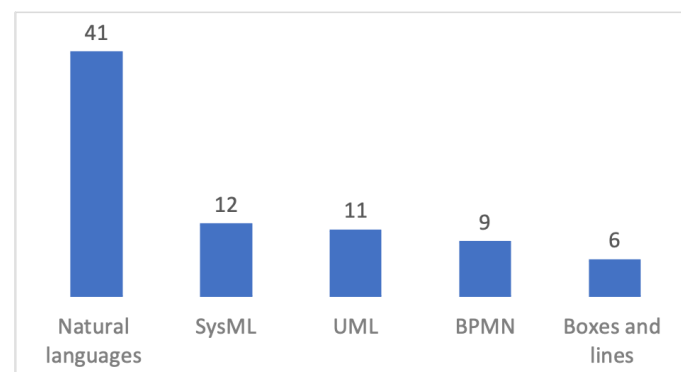
Figure 3. The requirements specification techniques supported by the requirements engineering tools.

A total of 25% of the requirements engineering tools given in Table 5 support the precise specifications of requirements using well-known modeling languages, including SysML, UML, and BPMN.

Table 6 shows the requirements engineering tools (36%) that support the analysis of the requirement specifications. Some of the tools that support the requirements analysis require practitioners to use modeling languages that lead to the precise requirement specifications for facilitating the analysability. Those tools are Cameo, Capella, Eclipse Papyrus, Enterprise Architect, Rhapsody, MagicDraw, Innovator for Business Analysts, and Visual Paradigm. The rest of the tools offer the use of the natural languages in some structured format and, thus, enable the analysis of the specifications for some properties. Note that a few of those tools that support the requirement specifications in natural languages (i.e., innoslate, ReqView, SwiftKanban, and Visure Requirements) use artificial intelligence and natural language processing technologies to check the informal requirements in natural languages (e.g., determining ambiguous requirements and similar requirements).

Table 5. The requirements engineering tools that support modeling languages for the requirements specification.

Requirements Engineering Tools	Modeling Languages
Business Optix	BPMN
Cameo Systems Modeler	SysML
Capella	SysML
CaseComplete	UML
Cradle	SysML
Eclipse (IDE) Papyrus	UML, SySML
Rational Rhapsody	UML, SySML
innoslate	LML, SysML
Innovator for Business Analysts	ArchiMate, BPMN, SysML
in-STEP BLUE	UML, SySML, and natural languages
MagicDraw	UML, SysML, BPMN, OWL, OCL, MARTE, SOAML
Visual Paradigm	UML, BPMN, ArchiMate, DFD, ERD, SoaML, SysML, CMMN
CA Agile Requirements Designer	Flowchart
TopTeam	UML, SysML, BPMN

Table 6. The requirements engineering tools that support the requirements analysis.

Requirements Engineering Tools	Pre-Defined Properties	User-Defined Properties	Simulation	Consistency	Completeness
agosense.fidelia	X				
Auros IQ	X				X
Business Optix			X		X
Cameo Systems Modeler	X	X	X	X	X
Capella	X	X	X		X
Cradle	X	X		X	X
Eclipse (IDE) Papyrus			X		
Enterprise Architect	X	X	X		
Rational DOORS					X
Rational Rhapsody	X	X	X		
innoslate	X		X	X	X
Innovator for Business Analysts	X				
MagicDraw	X	X	X		X
ReqEdit	X				
ReqView				X	
SwiftKanban				X	
Valispace				X	X
Visual Paradigm			X		
Visure Requirements	X			X	
ReqChecker	X			X	X

Checking requirements for predefined properties is the most popular feature, satisfied by many of the tools. Also, some of the tools that support predefined properties further enable users to define their own properties. Model simulation support is provided by the tools that essentially enable the precise specifications of the behaviour requirements via some modeling languages (e.g., UML, SysML, and BPMN). The completeness and

consistency checks for the requirements are also quite popular. Lastly, Cameo Systems Modeler is the only tool that supports all the analysis properties that are considered.

Table 7 shows that 75% of the requirements engineering tools support the requirements transformation in terms of any of the three features which are code generation, test scenario generation, and document generation. Thus, apparently, almost all of those tools support generating documents from requirement specifications in different formats such as Word, Excel, PDF, and HTML. Code generation from requirement specifications is supported by a few tools only, which are Cameo, Papyrus, Enterprise Architect, MagicDraw, Modelio, and Visual Paradigm. Note that those tools that support code generation are all software modeling and design tools that generate code from precise models specified with modeling languages (e.g., SysML and UML). Test scenario generation from the requirement specifications is supported by a small amount of tools, which are Quality Center—Dimensions RM, Auros IQ, RMsis, SpiraTeam, TopTeam, and CA Agile Requirements Designers.

Table 7. The requirements engineering tools that support the requirements transformation.

Requirements Engineering Tools	Code Generation	Test Scenario Generation	Document Generation
Aligned Elements			X
Quality Center-Dimensions RM		X	
Auros IQ		X	
Axosoft			X
Azure DevOps			X
Cameo Systems Modeler	X		
Capella			X
CaseComplete			X
codeBeamer ALM			X
Cradle			X
Eclipse (IDE) Papyrus	X		X
Enterprise Architect	X		X
Helix RM			X
Rational DOORS			X
innoslate			X
in-STEP BLUE			X
iRise			X
Kovair ALM			X
MagicDraw	X		X
Matrix ALM/QMS			X
Modelio Analyst	X		X
OpenProject			X
Polarion Requirements			X
Psoda			X
ReqEdit			X
ReQtest			X
ReqView			X
RMsis		X	X
SpiraTeam		X	X
Tuleap Enterprise			X
Valispace			X
Visual Paradigm	X		X

Table 7. *Cont.*

Requirements Engineering Tools	Code Generation	Test Scenario Generation	Document Generation
Visure Requirements			X
Yodiz			X
Accompa PM			X
CA Agile Requirements Designer		X	X
Doc Sheets			X
Orcanos			X
TopTeam		X	X
Xebrio			X
ReqChecker			X
StoriesOnBoard			X

3.4. Customisation

Customisation is considered in terms of the support for user-defined modeling viewpoints, integration with external tools, API support, and DSL support.

Concerning the user-defined modeling viewpoint, Cameo Systems Modeler, Enterprise Architect, Rational Rhapsody, MagicDraw, Modelio Analyst, and Visual Paradigm are the only tools that enable the users to define their own custom modeling viewpoints for specifying requirements. With those tools, users can reuse and extend the existing concepts from popular modeling languages (e.g., UML and SysML) and define rules and constraints on those concepts. By doing so, users can later consider the viewpoints to address their particular needs and concerns.

Concerning the integration with external tools, most of the requirements engineering tools (87%) support the integration with different types of tools, including test automation tools, project management tools, versioning and repository tools. The exceptions here are Modelio Analyst, OpenProject, ReqEdit, ReqView, RMTrack, CA Agile Requirements Designer, and TopTeam.

Concerning the API support, 70% of the requirements engineering tools offer an API for the users to extend the tools with specific features. Those tools provide websites that guide the user on how to use the API and perform any extensions. Figure 4 shows the tools that offer APIs and those that do not.
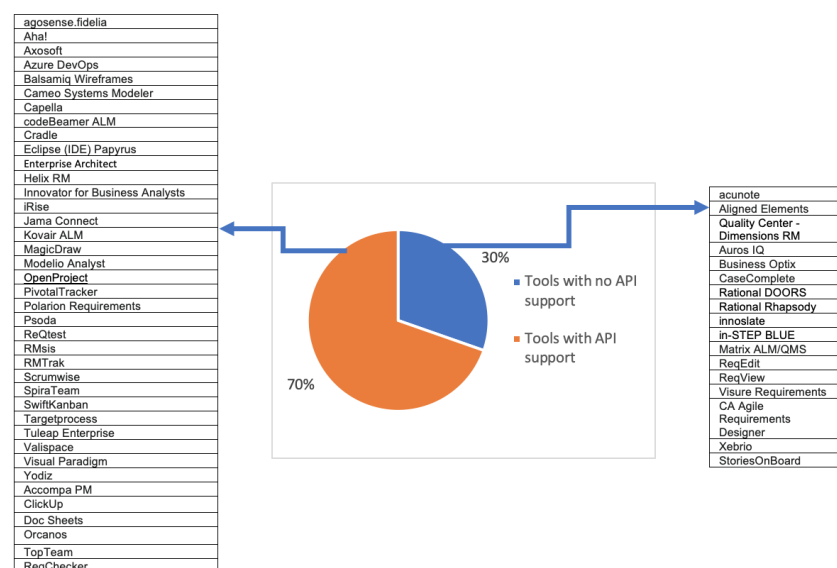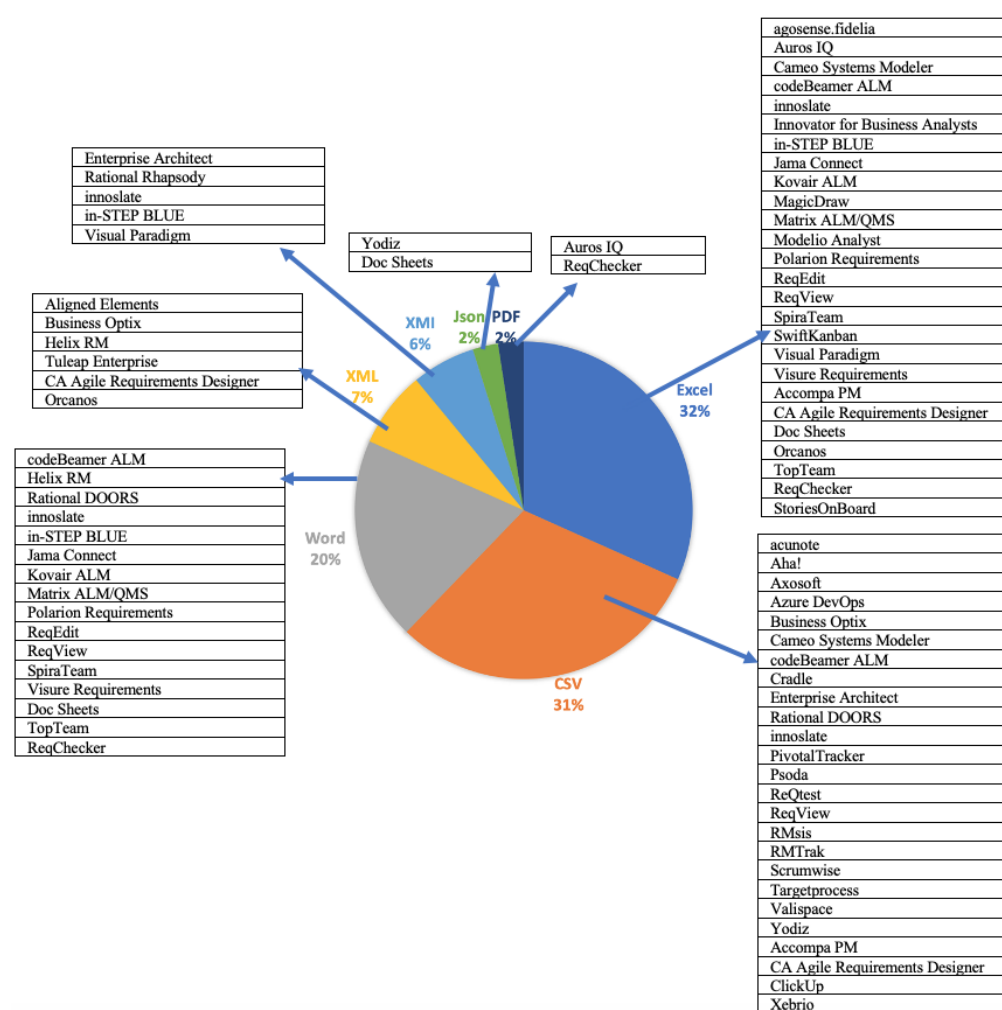

Figure 4. The requirements engineering tools that support API for enabling extension.

Concerning the DSL development support, most of the requirements engineering tools do not support the development of domain-specific languages for specifying requirements. The only exceptions here are some of the tools that support UML—Cameo Systems Modeler, Eclipse Papyrus, Enterprise Architect, Rational Rhapsody, MagicDraw, Modelio Analyst, and Visual Paradigm. Those tools that support UML modeling enable users to benefit from UML's profiling mechanism for extending UML with domain-specific concepts.

3.5. Interoperability

Interoperability is considered in terms of the support for importing/exporting requirements in different data formats.

Most of the requirements engineering tools (89%) import requirements in different data formats. The top-supported data formats are Excel (32%), CSV (31%), and Word (20%), which are followed by other formats, including XML, XMI, Json, and PDF. Figure 5 shows the tools that support the different data formats for importing requirements.

Figure 5. The requirements engineering tools that import the requirement specifications in different formats.

A total of 77% of the requirements engineering tools enable both importing and exporting requirements specifications in some data formats. The top-supported data format is ReqIF (25%), which is followed by Excel (23%) and CSV (23%). Figure 6 shows the tools that support different data formats for importing/exporting requirements.
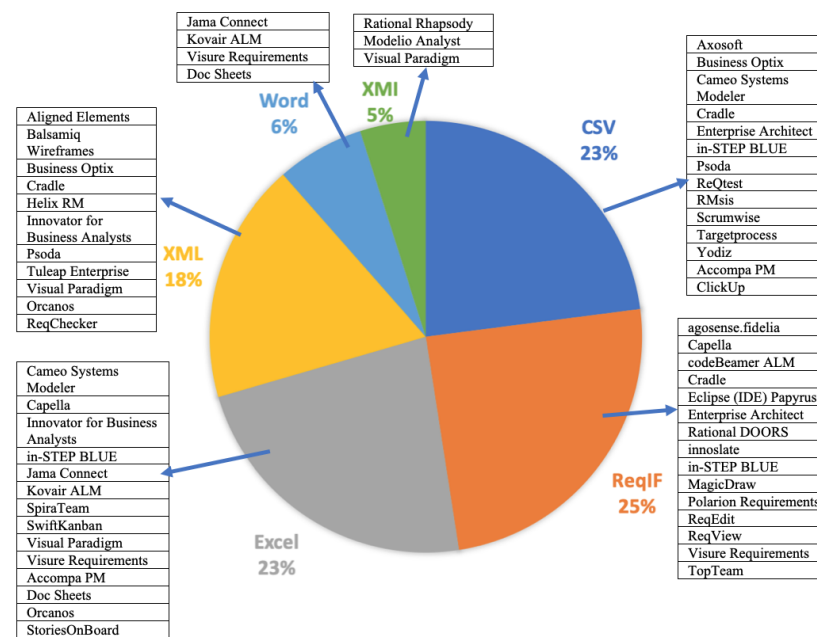
Figure 6. The requirements engineering tools that support importing and exporting requirement specifications in different formats.

3.6. Methodology

We consider three widely-used software development methodologies, which are model-driven engineering, agile software development, and product-line engineering. A total of 70% of the requirements engineering tools support at least one of those three methodologies, and those tools are given in Table 8. Agile software development is highly popular among the requirements engineering tools (52%), enabling the management of requirements using the agile development principles. A total of 32% of the tools support model-driven engineering and enable the specifications of models using modeling languages and performing other facilities such as model validation, model simulation, and model transformation (e.g., generating code from models). Product-line engineering is rarely supported in comparison with agile software development and model-driven engineering—Cameo Systems Modeler, codeBeamer ALM, and MagicDraw are the only tools that support the specifications of software systems with the principles of product-line engineering. Note that none of the requirements engineering tools support all the three methodologies at the same time.

Table 8. The requirements engineering tools that support different methodologies.

Requirements Engineering Tools	Model-Driven Engineering	Agile Software Development	Product-Line Engineering
acunote	No	Yes	No
agosense.fidelia	No	Yes	No
Aha!	No	Yes	No
Quality Center—Dimensions RM	Model-driven testing	Yes	No
Auros IQ	No	Yes	No
Axosoft	No	Yes	No
Azure DevOps	No	Yes	No
Business Optix	Model specification and simulation	Yes	No
Cameo Systems Modeler	Model specification and validation	No	Yes
Capella	Model specification and validation	No	No
CaseComplete	Model specification	No	No

Table 8. *Cont.*

Requirements Engineering Tools	Model-Driven Engineering	Agile Software Development	Product-Line Engineering
codeBeamer ALM	No	Yes	Yes
Cradle	Model specification and validation	Yes	No
Eclipse (IDE) Papyrus	Model specification and simulation	No	No
Enterprise Architect	Model specification, validation, and transformation	Yes	No
Rational Rhapsody	Model specification, validation, simulation, and transformation	No	No
innoslate	Model specification and simulation	No	No
Innovator for Business Analysts	Model specification	No	No
iRise	Model specification	Yes	No
Kovair ALM	Model specification	Yes	No
MagicDraw	Model specification, validation, and transformation	No	Yes
Modelio Analyst	Model specification, validation, and transformation	No	No
OpenProject	No	Yes	No
PivotalTracker	No	Yes	No
Psoda	No	Yes	No
ReQtest	No	Yes	No
RMsis	No	Yes	No
Scrumwise	No	Yes	No
SpiraTeam	No	Yes	No
SwiftKanban	No	Yes	No
Targetprocess	No	Yes	No
Tuleap Enterprise	No	Yes	No
Visual Paradigm	Model specification, validation, simulation, and transformation	Yes	No
Visure Requirements	No	Yes	No
Yodiz	No	Yes	No
Accompa PM	No	Yes	No
CA Agile Requirements Designer	Model-driven testing	No	No
ClickUp	No	Yes	No
Doc Sheets	No	Yes	No
TopTeam	Model-driven testing	No	No

3.7. User-Support

We consider the user-support viewpoint in terms of the requirements engineering tools' support for telephone communication, e-mail communication, forum, live-chat, help-desk, user guide, blog, white papers, mailing list, case-studies, videos, training, and coaching/consulting. Table 9 shows the related support provided by the requirements engineering tools. Thus, apparently, each tool enables users to access any tool-related knowledge to some extent. Indeed, each tool provides a user-guide for the users to learn how to use the tool. Likewise, most of the tools (73–77%) provide telephone details and e-mail addresses through which users may ask questions. A total of 73% of the tools provide videos for the users to see and learn how to use the tool. Help-desk, training support, and blog resources are also quite popular among the tools (50–57%); however, live-chat, mailing list, and coaching are rarely supported (7–27%).

Table 9. The user-support facilities provided by the requirements engineering tools.

Requirements Engineering Tools	Tel.	E-mail	Forum	Livechat	Help Desk	Guide	Blog	White Paper	M. List	Case Study	Videos	Training	Coaching
acunote	X	X				X	X		X				
agosense.fidelia	X	X		X		X	X						
Aha!	X	X	X			X	X						
Aligned Elements	X	X				X	X						
Quality Center—Dimensions RM	X	X	X		X	X	X	X	X			X	
Auros IQ	X	X				X	X						
Axosoft		X				X	X						
Azure DevOps			X			X	X						
Balsamiq Wireframes	X	X	X		X	X		X			X		
Business Optix	X	X			X	X		X		X	X	X	
Cameo Systems Modeler	X		X		X	X				X	X		X
Capella			X			X		X	X	X	X	X	X
CaseComplete	X	X			X	X					X		
codeBeamer ALM		X				X	X	X		X	X	X	X
Cradle	X	X				X	X	X			X	X	
Eclipse (IDE) Papyrus		X	X		X	X			X	X	X		
Enterprise Architect	X	X	X		X	X		X		X	X	X	X
Helix RM	X	X				X				X	X	X	X
Rational DOORS	X		X		X	X	X	X		X	X	X	X
Rational Rhapsody	X		X		X	X	X	X		X	X	X	X
innoslate	X	X		X	X	X	X				X	X	
Innovator for Business Analysts	X	X				X					X	X	
in-STEP BLUE	X	X				X	X	X			X		
iRise						X		X			X		

Table 9. *Cont.*

Requirements Engineering Tools	Tel.	E-mail	Forum	Livechat	Help Desk	Guide	Blog	White Paper	M. List	Case Study	Videos	Training	Coaching
Jama Connect	X		X		X	X	X				X		
Kovair ALM	X	X			X	X		X		X	X		
MagicDraw	X	X	X		X	X				X	X		X
Matrix ALM/QMS	X			X	X	X	X				X		
Modelio Analyst	X	X	X			X		X		X	X	X	X
OpenProject	X	X	X			X	X				X	X	X
PivotalTracker		X				X	X				X		X
Polarion Requirements	X		X	X		X		X			X	X	X
Psoda	X	X				X	X			X	X	X	
ReqEdit	X	X			X	X					X		
ReQtest		X				X	X					X	
ReqView	X				X	X				X	X	X	X
RMsis	X	X			X	X	X				X		
RMTrak	X	X			X	X					X		
Scrumwise		X		X		X							
SpiraTeam	X	X	X		X	X					X	X	
SwiftKanban	X	X				X	X			X	X	X	X
Targetprocess	X	X	X	X	X	X	X	X		X	X	X	
Tuleap Enterprise	X					X	X	X		X	X	X	X
Valispace		X		X	X	X	X	X		X	X		
Visual Paradigm	X	X	X		X	X	X	X		X	X	X	
Visure Requirements	X	X			X	X	X	X			X	X	
Yodiz		X	X		X	X	X				X		
Accompa	X	X				X	X			X	X		
CA Agile Requirements Designer	X	X	X	X	X	X			X		X	X	

Table 9. *Cont.*

Requirements Engineering Tools	Tel.	E-mail	Forum	Livechat	Help Desk	Guide	Blog	White Paper	M. List	Case Study	Videos	Training	Coaching
ClickUp				X	X	X		X		X		X	
Doc Sheets	X	X			X	X	X					X	
Orcanos	X	X	X	X	X	X	X			X	X	X	
TopTeam	X	X				X							
Xebrio		X			X	X		X			X		
ReqChecker					X	X						X	
StoriesOnBoard		X				X	X	X		X			

Enterprise Architect, Rational Rhapsody, Targetprocess, and Visual Paradigm are the tools that support the greatest number of criteria that are considered for the tool-support. Note that none of the tools support all the features that are considered here.

4. Discussion

4.1. Summary of Findings

In Section 3, we analysed 56 different requirements engineering tools for a number of features that are categorised as the project management, specification, collaboration, customisation, interoperability, methodology, and user-support viewpoints. In the remainder of this section, we summarise the key findings from our analysis.

The project management activities are not the priority for many tools. Only 46% of the requirements engineering tools provide built-in support for managing projects in terms of the project initiation, project planning, and project execution activities.

Multi-user collaboration support is provided by almost all the tools. All the requirements engineering tools support multi-user access—except ReqChecker. A total of 78% of those tools further enable the assigning of different roles to users (e.g., editor, reader, developer, tester, manager, etc.) and the configuration of access rights for the user roles.

Most of the tools provide their built-in versioning control system. A total of 74% of the requirements engineering tools provide built-in versioning systems, while the support for external versioning systems (e.g., GIT, SVN, and Mercurial) remains very low.

The most popular requirement specification technique is the natural language. A total of 73% of the requirements engineering tools support the requirements to be specified in natural languages rather than any modeling languages based on precise definitions.

The precise specification of requirements that can easily be processed is rarely supported. A total of 25% of the requirements engineering tools that support software modeling and design enable users to specify requirements precisely using well-accepted modeling languages such as SysML, UML, and BPMN. Using those tools, it is possible to specify requirements precisely, analyse requirements, and further transform the requirements.

Analysing requirement specifications to detect issues is rarely supported. Only 30% of the requirements engineering tools analyse the requirement specifications. While some of those tools enable the model analysis thanks to their support for the precise modeling languages, some tools offer the use of structured natural languages for the requirements specification and analysis. Note that a few tools use artificial intelligence and natural language processing techniques to check the informal requirement specifications in natural languages.

Requirements transformation is mainly considered for generating documents from requirements. A total of 75% of the requirements engineering tools support the requirement transformation, and most of those tools only support generating documents in formats such as Word, Excel, HTML, and PDF. However, generating skeleton code and test-scenarios from requirements are rarely supported.

Tool customisation is highly popular by means of the external tool integrations and API support. A total of 87% of the requirements engineering tools support the integration with many external tools, including the test automation tools, project management tools, and versioning tools. Also, 70% of the tools provide their own APIs through which users can develop their own tool extensions.

Extending the notation set for the requirement specifications is rarely supported. While the tool customisations/extensions are supported by most of the requirements engineering tools, very few of them are capable of extending the notation set for giving the domain-specific requirements (e.g., defining modeling viewpoints and developing DSLs).

Most tools accept requirements in different formats, and Excel/CSV are the most popular formats. A total of 89% of the requirements engineering tools enable the importation of requirements in different data formats including Excel, CSV, Word, XMI, Json, and PDF. The top import formats are Excel and CSV (31–32%), followed by Word (20%).

Many tools support importing and exporting the requirement specifications via the ReqIF, Excel, and CSV data exchange formats. A total of 77% of the requirements engineering tools enable exchanging (i.e., importing and exporting) of the requirement specifications. The top data exchange formats are ReqIF (25%), Excel (23%), and CSV (23%).

The top-supported software development methodology is agile. A total of 52% of the requirements engineering tools enable users to manage their requirements using agile principles and techniques. Model-driven engineering is supported by 32% of the requirements engineering tools, and product-line engineering is supported by just three tools.

User-guide, telephone details, e-mail addresses, and videos are the four most popular methods adopted by the tools for user-support. While all the requirements engineering tools provide user-guide documents, 73–77% of the tools provide telephone details, e-mail addresses, and videos.

4.2. Lessons Learned

Many requirements engineering tools promote the requirements to be specified in natural languages. While using natural languages is important for reducing the learning curve and enabling even nontechnical users to specify their requirements easily and quickly, processing the natural language specifications can be hard. With the use of natural languages, the requirements engineering tools essentially focus on the requirements gathering and documentation rather than the requirements analysis and transformation. Indeed, the natural language processing techniques and technologies are rarely adopted by the tool vendors for the requirements analysis. Note that we also observed a few tools that promote the specifications of requirements using the structured natural languages and therefore enable the requirements analysis.

The tools with software modeling and design support (e.g., Cameo, Visual Paradigm, Magic Draw, and Enterprise Architect) do enable the automated analysis and simulation of requirement specifications and their transformation into skeleton code. However, those tools require users to use modeling languages with precise definitions. Note that while the precision here facilitates the analysability and transformation of models, nontechnical users may not find it easy to learn and use the modeling languages for their requirement specifications.

While the requirements engineering tools enable the tool extensions via external tool integrations and API support, most of the tools ignore extending the notation sets used for specifying requirements (e.g., enabling DSL development). Only a small number of tools enable users to extend the well-known UML language via UML's profiling mechanism and define a domain-specific notation set. However, it should be noted that domain-specific modeling is highly important in increasing the productivity and maximising the quality of requirements engineering [41]. Indeed, many industries develop their own domain-specific languages using meta-modeling technologies [42,43].

We also learned that most of the requirements engineering tools support importing and exporting the requirements specifications. However, no single standard (e.g., ReqIF) is adopted by all of the tools for the data exchange. Also, none of the tools share any case studies that demonstrate exchange of the requirement specifications between different tools. Another interesting lesson is that while ReqIF is considered a well-regarded standard for exchanging requirements, only 25% of the requirements engineering tools support ReqIF. The rest of the requirements engineering tools support some other formats, including Excel, CSV, and XML.

Most of the requirements engineering tools enable the application of the agile principles and benefit from such techniques as Scrum, Kanban, and lean. While the agile techniques help in managing the projects effectively and developing software systems that meet the customer needs, agile development does not essentially address the important principles of software engineering such as reusability, maintainability, automated code (or test scenario) generation, and early detection of errors. Unfortunately, such methodologies

as product line engineering and model-driven engineering that are believed to enhance the software quality are rarely supported by the existing tools.

Software development lifecycle starts with the requirements engineering and continues with other processes to be performed, including design, implementation, and testing under the guidance of some process models (e.g., waterfall, V-model, incremental model, spiral model, etc.) [44]. To develop the right systems correctly, the artefacts produced in the requirements should be linked with the artefacts produced in design and implementation, which enable tracing from requirements to the implementation (and vice versa). While there exist tools that support the gradual development of software systems from requirement specifications to the code generation, the existing tools do not support the full round-trip engineering.

Another lesson is related to the user-support, as many of the requirements engineering tools provide inadequate support for the users to resolve their tool-related concerns. Indeed, given 13 different criteria that are considered for the user-support, more than half of the tools support 6 of those criteria at most, and, therefore, those tools may require some learning curve for the users as the users may not easily figure out how to use the tools and their particular features as they wish.

Among the 56 different requirements engineering tools, a few tools were observed to support the viewpoints and their features in a wider sense. Those tools are Enterprise Architect, MagicDraw, and Visual Paradigm, which essentially support all the viewpoints and most of their features. All the three tools here support the specifications of requirements using de facto modeling languages (e.g., UML, SysML, and BPMN) in a way that can automatically be analysed for the predefined and user-defined properties and further transformed into code and some useful documentation. All the three tools support the tool customisation by enabling the users to define their own requirement specification notation sets via user-defined viewpoints. The three tools all further support the collaborative modeling, project management facilities, and built-in versioning system. Also, while MagicDraw and Enterprise Architect support importing and exporting the requirement specifications in the standard ReqIF format, Visual Paradigm supports XML exclusively.

4.3. Threats to Validity

Internal validity is concerned with causal relationships between the results of the analysis and any independent variable (i.e., cause) that leads to the results [45]. In this study, nonprobabilistic sampling was used and the requirements engineering tools were chosen nonrandomly. That is, the tools to be analysed were sought from the Internet systematically and filtered according to the exclusion criteria as discussed in Section 2.3. In addition, the tools were analysed for various requirements engineering features categorized in seven viewpoints. Derivation and the formalization of these features and composing viewpoints were based on our previous research on understanding practitioners' experiences in requirements engineering [13].

Three experienced researchers were involved in the analysis of the requirements engineering tools. To minimise any instrumentation biases here, we ensured that the three researchers analysed the same set of tools independently of each other in the same systematic way discussed in Section 2.5. The results obtained from the researchers were compared to detect any inconsistencies in the data analysis.

External validity threats concern the generalizability of the analysis results, that is, the degree to which the examined studies are representative of the reviewed topic [45]. The set of requirements engineering tools analysed in our study may not be representative of the entire set of all available tools; however, this threat was mitigated by an extensive search on the Internet using various keywords, as listed in Section 2.3.

Construct validity relates to how well an analysis helps in achieving the research objective. Our goal was to analyse the capabilities of the existing requirements engineering tools according to the practitioners' needs. For this purpose, we categorized various requirement engineering features under seven viewpoints and used them during our

analysis. The analysed data were compared, existing inconsistencies were determined, and these parts were reanalysed together with all authors until they reached a consensus on them. This method also contributed to minimizing the risk on the construct validity of the conducted research. Additionally, we needed to ensure that all relevant requirement engineering tools were found adequately. For this purpose, well-known terms/concepts related to the requirement engineering tools and platforms were used to create search strings; several search iterations were provided and, hence, the adequate coverage of the all available tools was achieved.

Finally, to minimize the conclusion validity threat, the research methodology of this study was designed and validated carefully to minimize the risk of excluding relevant requirement engineering tools. Benefiting from our previous experience in conducting other analysis studies (e.g., [46]), the search methodology here was formalized and applied in a way such that only a very small number of relevant requirement engineering tools could be missed, and a manageable quantity of irrelevant ones could be included. Furthermore, the findings of the performed analysis were assessed within the context of the set of tool features provided at the beginning of the study.

5. Related Work

In this section, we discuss the similar studies that compare a set of requirements engineering tools for a number of features.

In [47], the authors aimed to provide a guidance for the practitioners on improving their requirement specifications and choosing the right tool(s) to manage the requirements. The authors analysed 13 important requirements engineering tools by observing their practical use in client environments. The authors essentially considered the agile methodologies, collaboration, and test-driven requirements engineering. The authors also provided interesting guidance on how to use the requirements engineering tools.

Taking into account the security support, seven requirement engineering tools were analysed in [48]. Covered features for the analysis were methodology (e.g., model-driven engineering, goal-oriented approach, Secure Tropos), the source of security element, the requirement formality, and the support for the requirements engineering activities (i.e., elicitation, specification, analysis, and verification). The authors essentially pointed out the tools' weak points in terms of gathering and documenting security requirements in a precise way that can be performed by stakeholders with limited technical knowledge and further validated.

In [49,50], the main motivation was to understand the capabilities of the requirement engineering tools existing so far. The authors surveyed the vendors of 38 different requirements engineering tools. The survey consisted of 146 questions and, with those questions, the tool vendors were expected to rank their tools support for different capabilities. Moreover, the authors performed three separate scenarios to better understand the particular features that are more important for practitioners and the tools' support for those features.

A total of 58 different UML modeling tools were examined in [14] according to the features (i.e., modeling viewpoints, analysis, transformation and export, collaboration, tool integration, scripting, project management, and knowledge management) that are believed to be important for practitioners. Note, however, that the author here focused on analysing the tools with UML support rather than the tools that support requirements management.

In [51], the authors aimed to analyse 21 different requirements engineering tools for a number of functionality features and their geographical locations and understand the popular tool features and the countries with greater support for the requirements engineering tools. The functionality features were elicitation, specification, analysis, verification, traceability, documentation, graphical representation and tool integration. The authors simply indicated the tools' support for those features using the yes/no answer style without giving thorough discussions of the features.

The main objective of the analysis made in [52] was the tool support on the requirements management and traceability. Thirteen different requirements engineering tools

were analysed, considering features including the automatic link detection, automatic link creation/change, coverage analysis support, documentation support, graphical representation, and tool integration. The authors simply indicated which tools support which of the features using yes/no answers.

In [53], the authors aimed to perform the qualitative analysis of 12 different requirements engineering tools via a number of features and compare the tools for some organisational factors. The features considered here included traceability, analysis, security and accessibility, portability, configuration management, collaboration, change management, usability, and specification. Moreover, the organisational factors included cost, licensing fees, and platform requirements.

The review conducted in [54] aimed at analysing 10 requirements engineering tools with regard to their support for the artificial-intelligence-based requirements gathering and requirements management techniques and activities. However, support of the examined tools according to these features were given only with yes/no style answers (i.e., *low*, *medium*, and *high*) without any thorough discussions of the analysis results or lessons learned.

In [55], the authors aimed to analyse eight different requirements engineering tools, categorised as heavyweight, middleweight, and lightweight tools. The authors focused on a set of features (i.e., traceability, template, graphical representation, tool integration, scalability, and glossary) and analysed each tool to understand how many of those features were supported.

In [56], the authors conducted a survey among 117 students to understand their thoughts on nine highly-used UML modeling tools. The authors focused on understanding, from the perspectives of the students, the main benefits and drawbacks of the tools.

In [57], the authors focused on three requirements engineering tools that support the natural language processing and aimed to analyse and compare the three tools for ambiguity and atomicity. The authors also proposed a framework for a next-generation requirements engineering tool that combines the strengths of the existing tools in terms of the natural language processing.

In [58], the authors aimed to determine the features for analysing the product-line software engineering tools and analyse the existing tools for those features. The authors selected four different tools that support the product-line software engineering and analysed them with regard to three viewpoints (i.e., product-line engineering, management, and technical) which led to 13 different features in total.

Tables 10 and 11 give the comparisons of the review studies that are discussed above. In Table 10, the review studies are compared with regard to *(i)* their publication year, *(ii)* the time period of the tool versions considered, *(iii)* the number of tools and features considered, *(iv)* the review type (i.e., systematic or not), and *(v)* the search strings used to reach the tools. Note that when any information was not reached, it is indicated as "Not given". In Table 11, the review studies are compared with regard to the support for the feature viewpoints that are considered in our study. Therefore, none of the existing literature review studies analysed such a great number of requirements engineering tools for a diverse set of features in as systematic a way as in our study. Most studies that are considered here are literature review studies that do not perform any systematic reviews of the tools. In addition, our study is distinguished by its consideration of a comprehensive set of tools that are analysed for a comprehensive set of features which are grouped into multiple viewpoints. It should also be noted that, unlike our study, many of the existing studies focus more on understanding the tools that support certain features without collecting and analysing any precise data about those features. For instance, the support for the requirements transformation could be considered more precisely if the requirements transformation was addressed in terms of code generation, test-scenario generation, and document generation, as in our paper. Moreover, some studies just focus on the requirements engineering tools in particular domains rather than considering the tools in general. Our study is further distinguished by its consideration of the project management, collaboration, requirements

specification, customisation, interoperability, methodology, and user-support viewpoints, at the same time each of which is addressed with a cohesive set of features.

Table 10. The comparison of similar studies in the literature from different aspects.

Study	Publication Year	Period of the Tools	Number of the Tools	Number of the Features	Systematic Review	Search String Used
[47]	2021	2017–2021	13	17	No	Not given
[48]	2013	Not given	7	9	No	Not given
[49,50]	2012	Not given	38	23	Yes	Seven Databases used (Ian Alexander, Alarcos Research Group, INCOSE, Ludwig Consulting Services, Qaguild, Volere, and @WEBO)
[14]	2019	Not given	58	9	No	UML tool, UML toolset, UML editor, UML modeler, UML modeling tool, UML application, Online UML tool, UML tool free/commercial, UML for practitioners
[51]	2017	2002–2017	21	8	Yes	Software Engineering Tools, Elicitation Tools, and Requirement Management tool
[52]	2011	2001–2011	13	9	No	Not given
[53]	2003	Not given	12	13	No	Not given
[54]	2022	Not given	10	10	No	Not given
[55]	2022	Not given	8	7	No	Not given
[56]	2019	Not given	31	Student Survey	No	Not given
[57]	2016	2004–2015	3	2	Yes	Snowballing the literature
[58]	2007	2005–2007	4	13	Yes	Not given
Our Study	2023	1993–2017	56	20	Yes	see Section 2.2.3

Table 11. The comparison of similar studies with regard to the viewpoints focused on in our study.

Study	Project Man.	Specification	Collaboration	Customisation	Interoperability	Methodology	User-Support
[47]	No	No	Yes	No	Yes	Yes	Yes
[48]	No	Yes	No	No	No	Yes	No
[49,50]	Yes	Yes	No	No	Yes	No	No
[14]	Yes	Yes	Yes	Yes	Yes	No	Yes
[51]	No	Yes	No	Yes	No	No	No
[52]	No	Yes	No	Yes	No	No	No
[53]	No	Yes	Yes	No	Yes	No	No
[54]	No	Yes	Yes	No	No	No	No
[55]	No	Yes	Yes	Yes	No	No	No
[56]	No	No	No	No	No	No	No
[57]	No	No	No	No	No	No	No
[58]	No	Yes	Yes	No	Yes	No	No
Our Study	Yes	Yes	Yes	Yes	Yes	Yes	Yes

6. Conclusions

In this study, we analysed 56 different requirements engineering tools for a comprehensive set of features that were categorised into multiple viewpoints (i.e., project management, specification, collaboration, customisation, interoperability, methodology, and user-support). The analysis results revealed many important lessons. Most requirements engineering tools promote the requirements to be specified in natural languages. While using natural languages is important for reducing the learning curve, processing natural language specifications can be very hard (if not impossible). The requirements engineering tools enable the tool extensions via external tool integrations and API support; however, most of the tools ignore the extension of the notation sets used for specifying requirements. Most tools support the exchange of requirements data among different tools; however, no single standard (e.g., ReqIF) is adopted by all of the tools for the data exchange. Most tools support the agile principles and such techniques as Scrum, Kanban, and lean, while any other methodologies, such as product-line engineering and model-driven engineering, that can significantly improve the quality of software development are rarely supported. None of the tools enable round-trip engineering—i.e., receiving requirements in natural languages, specifying and verifying design models, linking requirements with the design models, and transforming all those into test-scenarios and code.

We strongly believe that the results of our literature review can be uniquely used as a reference guide for understanding and comparing the requirements engineering tools in general without being restricted to particular tools and domains. Unlike the existing studies in the literature, our study reveals *(i)* the greatest number of requirements engineering tools that can be used for specifying requirements in general and *(ii)* several interesting viewpoints that concern the requirements engineering and are explained by many different tool necessities. Many different stakeholders can, therefore, find the analysis results useful. Indeed, practitioners can use the results to compare the existing requirements engineering tools and find the ones that best meet their needs. Tool vendors can use the results to determine the gaps that can be improved. Researchers can conduct further empirical studies to better understand practitioners' perspectives and propose new projects that improve the existing requirements engineering tools.

In the future, our first goal is to conduct a series of interviews with a group of developers from diverse industries so as to receive their feedback about the analysis results and validate the lessons learned. We will also design a survey that primarily focuses on practitioners' challenges regarding requirements engineering tools.

Notes

[1] PRISMA 2020 checklist: http://www.prisma-statement.org/PRISMAStatement/Checklist (accessed on 15 April 2023.).
[2] PRISMA Flow Diagram: http://www.prisma-statement.org/PRISMAStatement/FlowDiagram (accessed on 15 April 2023).
[3] https://osate.org/ (accessed on 15 April 2023).
[4] The collected data can be accessed via the link: https://zenodo.org/records/10184749 (accessed on 22 November 2023).
[5] https://www.atlassian.com/software/jira (accessed on 9 April 2023).

References

1. *IEEE Std 1220-2005 (Revision of IEEE Std 1220-1998)*; IEEE Standard for Application and Management of the Systems Engineering Process. IEEE: Piscataway, NJ, USA, 2005; pp. 1–96. [CrossRef]
2. SEBoK Editorial Board (Ed.) *Guide to the Systems Engineering Body of Knowledge (SEBoK)*, version 2.9; 2023. Available online: https://sebokwiki.org/ (accessed on 1 April 2023).
3. Lethbridge, T.C.; Lagamiere, R. *Object-Oriented Software Engineering—Practical Software Development Using UML and Java*; MacGraw-Hill: New York, NY, USA, 2001.
4. Curcio, K.; Navarro, T.; Malucelli, A.; Reinehr, S.S. Requirements engineering: A systematic mapping study in agile software development. *J. Syst. Softw.* **2018**, *139*, 32–50. [CrossRef]
5. Nuseibeh, B.; Easterbrook, S. Requirements Engineering: A Roadmap. In Proceedings of the ICSE'00: Conference on the Future of Software Engineering, Limerick, Ireland, 4–11 June 2000; pp. 35–46. [CrossRef]
6. Laplante, P.A. *Requirements Engineering for Software and Systems*, 3rd ed.; Auerbach Publications: Sebastopol, CA, USA, 2017.
7. Pohl, K. *Requirements Engineering: Fundamentals, Principles, and Techniques*, 1st ed.; Springer Publishing Company, Incorporated: Berlin/Heidelberg, Germany, 2010.
8. Humphrey, W. Why Big Software Projects Fail: The 12 Key Questions. *J. Def. Softw. Eng.* **2005**, *18*, 25–29.
9. Charette, R. Why software fails [software failure]. *IEEE Spectrum* **2005**, *42*, 42–49. [CrossRef]
10. Hussain, A.; Mkpojiogu, E.O.C. Requirements: Towards an understanding on why software projects fail. *AIP Conf. Proc.* **2016**, *1761*, 020046. [CrossRef]
11. Ebert, C.; Jastram, M. ReqIF: Seamless Requirements Interchange Format between Business Partners. *IEEE Softw.* **2012**, *29*, 82–87. [CrossRef]
12. Page, M.J.; McKenzie, J.E.; Bossuyt, P.M.; Boutron, I.; Hoffmann, T.C.; Mulrow, C.D.; Shamseer, L.; Tetzlaff, J.M.; Akl, E.A.; Brennan, S.E.; et al. The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *Int. J. Surg.* **2021**, *88*, 105906. [CrossRef]
13. Ozkaya, M.; Akdur, D.; Toptani, E.C.; Kocak, B.; Kardas, G. Practitioners' Perspectives towards Requirements Engineering: A Survey. *Systems* **2023**, *11*, 65. [CrossRef]
14. Ozkaya, M. Are the UML modelling tools powerful enough for practitioners? A literature review. *IET Softw.* **2019**, *13*, 338–354. [CrossRef]
15. Ozkaya, M. The analysis of architectural languages for the needs of practitioners. *Softw. Pract. Exper.* **2018**, *48*, 985–1018. [CrossRef]
16. *ISO/IEC TR 24766:2009*; Information Technology—Systems and Software Engineering—Guide for Requirements Engineering Tool Capabilities. Technical Report ISO/IEC JTC 1/SC 7—Software and Systems Engineering; ISO/IEC: Geneva, Switzerland, 2009.
17. Shafiq, M.; Zhang, Q.; Akbar, M.A.; Khan, A.A.; Hussain, S.; Amin, F.E.; Khan, A.; Soofi, A.A. Effect of Project Management in Requirements Engineering and Requirements Change Management Processes for Global Software Development. *IEEE Access* **2018**, *6*, 25747–25763. [CrossRef]
18. Verner, J.; Cox, K.; Bleistein, S.; Cerpa, N. Requirements Engineering and Software Project Success: An industrial survey in Australia and the U.S. *Australas. J. Inf. Syst.* **2005**, *13*, 225-238. [CrossRef]
19. Arnaut, B.M.; Ferrari, D.B.; de Oliveira e Souza, M.L. A requirements engineering and management process in concept phase of complex systems. In Proceedings of the 2016 IEEE International Symposium on Systems Engineering (ISSE), Edinburgh, UK, 3–5 October 2016; pp. 1–6. [CrossRef]
20. Westland, J. *The Project Management Life Cycle: A Complete Step-By-Step Methodology for Initiating, Planning, Executing & Closing a Project Successfully*; Kogan Page, Limited: London, UK, 2006.
21. Taylor, R.N.; Medvidovic, N.; Dashofy, E.M. *Software Architecture—Foundations, Theory, and Practice*; Wiley: Hoboken, NJ, USA, 2010; pp. I–XXIV, 1–712.
22. Rumbaugh, J.E.; Jacobson, I.; Booch, G. *The Unified Modeling Language Reference Manual*; Addison-Wesley-Longman: Devon, UK, 1999; pp. I–XVII, 1–550.
23. Balmelli, L. An Overview of the Systems Modeling Language for Products and Systems Development. *J. Object Technol.* **2007**, *6*, 149–177. Available online: www.sysml.org (accessed on 1 April 2023). [CrossRef]
24. Völzer, H. An Overview of BPMN 2.0 and Its Potential Use. In Proceedings of the Business Process Modeling Notation—Second International Workshop, BPMN 2010, Potsdam, Germany, 13–14 October 2010; Proceedings; Mendling, J., Weidlich, M., Weske, M., Eds.; Lecture Notes in Business Information Processing; Springer: Berlin/Heidelberg, Germany, 2010; Volume 67, pp. 14–15. [CrossRef]
25. Feiler, P.H.; Gluch, D.P.; Hudak, J.J. *The Architecture Analysis & Design Language (AADL): An Introduction*; Technical Report; Software Engineering Institute: Pittsburgh, PA, USA, 2006.
26. Zowghi, D.; Gervasi, V. On the interplay between consistency, completeness, and correctness in requirements evolution. *Inf. Softw. Technol.* **2003**, *45*, 993–1009. [CrossRef]
27. Szyperski, C. Independently extensible systems-software engineering potential and challenges. *Aust. Comput. Sci. Commun.* **1996**, *18*, 203–212.
28. Lago, P.; Malavolta, I.; Muccini, H.; Pelliccione, P.; Tang, A. The Road Ahead for Architectural Languages. *IEEE Softw.* **2015**, *32*, 98–105. [CrossRef]

29. *IEEE Std 1471-2000*; IEEE Recommended Practice for Architectural Description for Software-Intensive Systems. IEEE: Piscataway, NJ, USA, 2000; pp. 1–30. [CrossRef]
30. Selić, B.; Gérard, S. Chapter 2—An Introduction to UML Profiles. In *Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE*; Selić, B., Gérard, S., Eds.; Morgan Kaufmann: Boston, MA, USA, 2014; pp. 27–43. [CrossRef]
31. Wegner, P. Interoperability. *ACM Comput. Surv. (CSUR)* **1996**, *28*, 285–287. [CrossRef]
32. Motta, R.C.; De Oliveira, K.M.; Travassos, G.H. Rethinking Interoperability in Contemporary Software Systems. In Proceedings of the 2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS), Buenos Aires, Argentina, 23–23 May 2017; pp. 9–15. [CrossRef]
33. Dingsøyr, T.; Nerur, S.; Balijepally, V.; Moe, N.B. A decade of agile methodologies: Towards explaining agile software development. *J. Syst. Softw.* **2012**, *85*, 1213–1221. [CrossRef]
34. Kent, S. Model Driven Engineering. In Proceedings of the Integrated Formal Methods, Third International Conference, IFM 2002, Turku, Finland, 15–18 May 2002; Proceedings; Butler, M.J., Petre, L., Sere, K., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2335, pp. 286–298. [CrossRef]
35. Metzger, A.; Pohl, K. Software product line engineering and variability management: achievements and challenges. In Proceedings of the Future of Software Engineering, FOSE 2014, Hyderabad, India, 31 May–7 June 2014; Herbsleb, J.D., Dwyer, M.B., Eds.; ACM: New York, NY, USA, 2014; pp. 70–84. [CrossRef]
36. Al-Zewairi, M.; Biltawi, M.; Etaiwi, W.; Shaout, A. Agile Software Development Methodologies: Survey of Surveys. *J. Comput. Commun.* **2017**, *5*, 74–97. [CrossRef]
37. Brambilla, M.; Cabot, J.; Wimmer, M. *Model-Driven Software Engineering in Practice*, 2nd ed.; Morgan & Claypool Publishers: Kentfield, CA, USA, 2017.
38. Thüm, T.; Apel, S.; Kästner, C.; Schaefer, I.; Saake, G. A Classification and Survey of Analysis Strategies for Software Product Lines. *ACM Comput. Surv.* **2014**, *47*, 6:1–6:45. [CrossRef]
39. Schön, E.; Thomaschewski, J.; Escalona, M.J. Agile Requirements Engineering: A systematic literature review. *Comput. Stand. Interfaces* **2017**, *49*, 79–91. [CrossRef]
40. Alves, V.; Niu, N.; Alves, C.F.; Valença, G. Requirements engineering for software product lines: A systematic literature review. *Inf. Softw. Technol.* **2010**, *52*, 806–820. [CrossRef]
41. Wasowski, A.; Berger, T. *Domain-Specific Languages—Effective Modeling, Automation, and Reuse*; Springer Cham: Berlin/Heidelberg, Germany, 2023. [CrossRef]
42. Kosar, T.; Bohra, S.; Mernik, M. Domain-Specific Languages: A Systematic Mapping Study. *Inf. Softw. Technol.* **2016**, *71*, 77–91. [CrossRef]
43. Leblebici, O.; Kardas, G.; Tuglular, T. A Domain-Specific Language for the Document-Based Model-Driven Engineering of Business Applications. *IEEE Access* **2022**, *10*, 104093–104110. [CrossRef]
44. Ruparelia, N.B. Software development lifecycle models. *ACM SIGSOFT Softw. Eng. Notes* **2010**, *35*, 8–13. [CrossRef]
45. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B. *Experimentation in Software Engineering*; Springer: Berlin/Heidelberg, Germany, 2012. [CrossRef]
46. Arslan, S.; Ozkaya, M.; Kardas, G. Modeling Languages for Internet of Things (IoT) Applications: A Comparative Analysis Study. *Mathematics* **2023**, *11*, 1263. [CrossRef]
47. de Gea, J.M.C.; Ebert, C.; Hosni, M.; Vizcaíno, A.; Nicolás, J.; Alemán, J.L.F. Requirements Engineering Tools: An Evaluation. *IEEE Softw.* **2021**, *38*, 17–24. [CrossRef]
48. Yahya, S.; Kamalrudin, M.; Sidek, S. A review on tool supports for security requirements engineering. In Proceedings of the 2013 IEEE Conference on Open Systems (ICOS), Kuching, Malaysia, 2–4 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 190–194. [CrossRef]
49. de Gea, J.M.C.; Nicolás, J.; Alemán, J.L.F.; Álvarez, J.A.T.; Ebert, C.; Vizcaíno, A. Requirements Engineering Tools. *IEEE Softw.* **2011**, *28*, 86–91. [CrossRef]
50. de Gea, J.M.C.; Nicolás, J.; Alemán, J.L.F.; Toval, A.; Ebert, C.; Vizcaíno, A. Requirements engineering tools: Capabilities, survey and assessment. *Inf. Softw. Technol.* **2012**, *54*, 1142–1157. [CrossRef]
51. Shah, A.; Alasow, M.A.; Sajjad, F.; Baig, J.J.A. An evaluation of software requirements tools. In Proceedings of the 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 5–7 December 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 278–283. [CrossRef]
52. Shahid, M.; Ibrahim, S.; Mahrin, M.N. An Evaluation of Requirements Management and Traceability Tools. *Int. J. Comput. Inf. Eng.* **2011**, *5*, 627–632.
53. Sud, R.R.; Arthur, J.D. *Requirements Management Tools: A Quantitative Assessment*; Technical Report TR-03-10; Department of Computer Science, Virginia Polytechnic Institute & State University: Blacksburg, VA, USA, 2003.
54. Nadeem, M.A.; Lee, S.U.J.; Younus, M.U. A Comparison of Recent Requirements Gathering and Management Tools in Requirements Engineering for IoT-Enabled Sustainable Cities. *Sustainability* **2022**, *14*, 2427. [CrossRef]
55. Inam-Ul-Haq.; Abbas, W.; Butt, W.H. Systematic Literature Review on Requirement Management Tools. In Proceedings of the 2022 International Conference on Emerging Trends in Smart Technologies (ICETST), Karachi, Pakistan, 23–24 September 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6. [CrossRef]

56. Agner, L.T.W.; Lethbridge, T.C.; Soares, I.W. Student experience with software modeling tools. *Softw. Syst. Model.* **2019**, *18*, 3025–3047. [CrossRef]

57. Arendse, B.; Lucassen, G. Toward Tool Mashups: Comparing and Combining NLP RE Tools. In Proceedings of the 24th IEEE International Requirements Engineering Conference, RE 2016, Beijing, China, 12–16 September 2016; IEEE Computer Society: Piscataway, NJ, USA, 2016; pp. 26–31. [CrossRef]

58. Djebbi, O.; Salinesi, C.; Fanmuy, G. Industry Survey of Product Lines Management Tools: Requirements, Qualities and Open Issues. In Proceedings of the 15th IEEE International Requirements Engineering Conference, RE 2007, New Delhi, India, 15–19 October 2007; IEEE Computer Society: Piscataway, NJ, USA, 2007; pp. 301–306. [CrossRef]