Requirements for Machine Learning Methodology Software Tooling

Jochen L. Leidner and Michael Reiche

1 Introduction

Over the course of the last two decades, there has been an enormous growth in the importance of data-intensive projects, projects aiming at obtaining data-driven insights ("analytics"), and components that apply automatic induction, also known as machine learning, instead of traditional algorithms, to solve a problem at hand. This is because there is a human desire to push the "how" question onto the machine for solving.

This move from algorithms to "soft computing" models induced from data also means that new methodologies are needed that recommend news processes (e.g., to annotate datasets) and new best practices (e.g., to compute inter-annotator agreement between annotators) to build such models. These (we will take a closer look below) extend our toolbox of software engineering methodologies (waterfall, agile kanban,[1] etc.) with methodologies suitable for and indeed specifically designed for machine learning based working, which is typically quantitative, iterative, and experimental.

[1] A kanban board is a software tool to support the kanban (Japanese for "billboard") methodology, which relies on two primary practices: 1. to visualize work and 2. to limit work in progress.

J. L. Leidner (✉)
Coburg University of Applied Sciences, Coburg, Germany

KnowledgeSpaces UG (haftungsbeschränkt), Coburg, Germany
e-mail: leidner@acm.org

M. Reiche
Coburg University of Applied Sciences and Arts, Coburg, Germany
e-mail: michael.reiche@hs-coburg.de

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

M. Mora et al. (eds.), *Development Methodologies for Big Data Analytics Systems*, Transactions on Computational Science and Computational Intelligence, https://doi.org/10.1007/978-3-031-40956-1_4

As we transition to this new breed of methodology, naturally we would like to make use of state-of-the-art software tools that support our methodology of choice. Where traditional software engineering gave us *computer-aided software engineering* (*CASE*) tools [2], we hope for equivalent guidance in the new, data-centric world.

To this end, we present a collection of requirements for such a software stack: In this chapter, we will gather and organize requirements for software tooling to support machine learning/data science methodologies.[2] For the most part, we will be able to defer the choice of methodology, as it turns out the software tooling requirements can be separated from any particular methodology.

Software engineering—and in the era of soft computing this includes construction of machine learning models—does not happen in isolation: stakeholders need to be educated, influenced, convinced and kept informed, developers briefed about interfaces and non-formalized aspects of integration and maintenance; even holders of financial roles need to learn that model refresh is a recurring post-project activity that needs funding and staffing, which may be unwelcome news in projects motivated by the "saving through automation" promised that machine learning offers. While machine learning researchers typically zoom in on only the mathematical or experimental work surrounding parameter estimation and evaluation of their models, real-life projects require extraordinary amounts of interactions with the environment. This requires new methodologies that are now beginning to emerge, and these in turn require software tooling to capture, store, process, retrieve, etc. the project knowledge that needs to be managed.

The remainder of this chapter is structured as follows: Sect. 2 provides some background about requirements and the requirements capture process. Section 3 briefly recapitulates an exemplary selection of methodologies for machine learning projects. Section 4 describes our collected requirements for methodology software tooling, including requirements from the perspective of what stakeholders need from the system to support their work. Section 5 surveys some related work. In Sect. 6, we provide a critical discussion. Finally, Sect. 7 summarizes our findings and concludes with suggestions for future work.

2 Method: From Stakeholders to Requirements Capture

In order to develop software tools to support the application of (and compliance with) machine learning methodologies that assist project teams and other stakeholders with the functionalities required in the realization of machine learning projects, suitable requirements must be identified, formulated in high quality, and documented in a structured way [3–5]. Stakeholders are individuals or organizations with an interest in the planned system [4, p. 10]. Typical team members and other stakeholders' responsibilities of a machine learning project and their tasks can be found in Table 1.

[2] In the following, the term "machine learning" will be used, although it is also intended to refer to the entire field of data science.

Table 1 Team members and other stakeholders and their typical responsibilities

Team members (top) and other stakeholders (bottom)	Responsibilities
Data scientist	Model engineering including data preparation, algorithm selection, hyperparameters selection, and model evaluation [6]
Data engineer	Data processing including data collection, feature engineering, big data management, data pipeline management, and dataset building [6, 7]
Software engineer/ machine learning engineer	Turn the raw machine learning problem into a prototype or a well-engineered product, monitoring of drifts and adjustments of those [6, 8]
Technical lead	Take charge of all technical decisions
Project manager (PM)	Administer, facilitate, manage: setting, updating, and monitoring team activities, project, and time plans; communication with stakeholders; defines the business goal; review the achievement of objectives; mediate issues and fights in the team, enabling innovation [6, 8]
Machine learning solution architect	Integration of machine learning into the IT infrastructure [8]
User experience specialist	Ensures interacting with the system will be intuitive, pleasant, and successful (free from obstacles) [8]
Subject-matter expert	Bringing understanding of a subject area [8]
User (focus group)	Provides feedback and requirements. "User" refers to an end user of the system – product or service – that the machine learning models are going to be part of
Executive sponsor	Making budget decisions, establishing a vision and several main goals of the project [8]

The IEEE defines a requirement as follows [9, p. 62]: "(1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2)" [9, p. 62].

Suitable requirements can be formulated generally at a higher level from a user perspective in natural language (user requirement) or in detail at a deeper level from a system perspective close to the software to be developed (system requirement) [5, 10]. Requirements do not provide information on how they are to be implemented. Rather, they provide information about what the system was intended to do [11, p. 230–231]. From the variants for classifying requirements, we use those that distinguish according to (1) priority; (2) necessity, i.e., must/should/will); and (3) functionality (nonfunctional/functional) [12, p. 80]. We use a template-based approach to formulate the requirements (see Fig. 1).

In principle, both analytical and empirical methods are suitable for the systematic eliciting of requirements. In Sect. 4, we elicit requirements for the needs of the typical stakeholders with the definition of characteristic user stories, which are derived from special properties of machine learning methodologies and from practical experience of the two authors of this chapter, respectively. From these, we extract a set of formal requirement templates (Fig. 1) and eventually entities that can form the basis for a class diagram (in an object-oriented analysis) or an ER (entity-relationship) diagram.

3 Machine Learning Process Models

In the following, we will recapitulate some selected methodologies starting with the oldest that have been proposed for projects using machine learning, data mining, and data science generally speaking. For detailed surveys, see the related work below.

3.1 KDD

The KDD process [14–16] resulted from the "Knowledge Discovery from Databases" community, which also created the conference series of the same name. Data mining, according to it, is a five-step process with iterations to get from raw data to knowledge. First, carrying out a data selection step results in target data, a preprocessing activity leads to preprocessed data, transformations lead to transformed data, and then the actual data mining turns it into patterns, which are interpreted by humans and/or evaluated by machine and/or humans (Fig. 2).

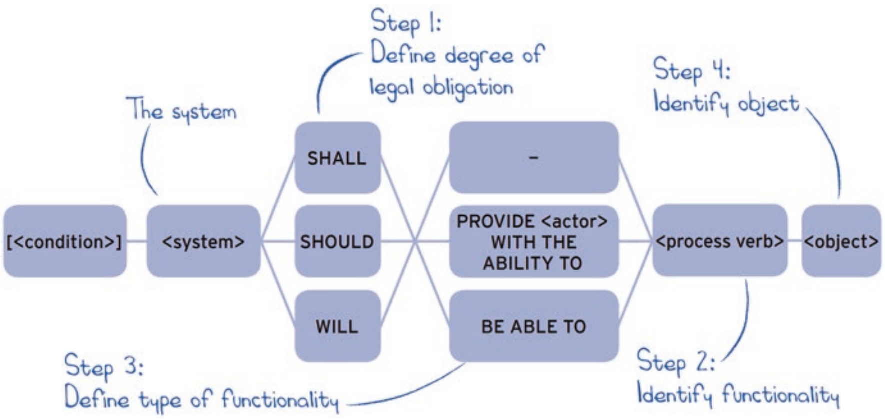

Fig. 1 Requirement template [13]

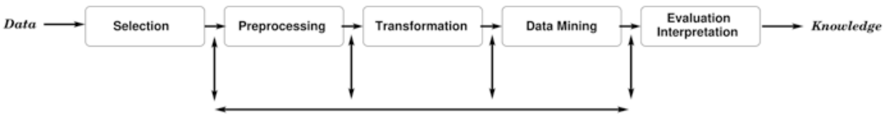

Fig. 2 The KDD process (simplified after Fig. 1 from [15, p. 29])

KDD starts in its first phase with the selecting a dataset (database, set of variables or data sample under consideration). Preprocessing work then removes noise/outliers, plugs gaps by imputation, and dealing with database schemas falls into this step. After that, the data is transformed (reduced and/or projected), which includes finding and extracting features that are useful for the task at hand and reducing its complexity. Next, the data mining algorithms/methods have to be selected (models, parameters). Framing or choosing the function of the data mining step then involves reflecting on the purpose of the model (e.g., summarization, classification, regression or clustering). The actual data mining phase applies classification, regression, some form of clustering, or sequence modeling suitable for the goals and dataset at hand. The final phase includes making sense of automatically discovered patterns, potentially using visualizations, weeding out superfluous or useless patterns, and translating useful patterns into language understandable by the project's stakeholders. Implicit, but accounted for in the methodology's prose descriptions, is also making use of the resulting knowledge gained, which often means integrating it into a software system, which the authors call "performance system"; this corresponds to the deployment phase in other methodologies. Documenting and reporting concludes a KDD process-based project.

3.2 SEMMA

The SEMMA methodology [17] was developed by SAS, Inc., a software company which also sells the SAS Enterprise Miner software, which is somewhat aligned with the SEMMA process. The process is divided into the following five phases:

- Sample: A subset of the appropriate data must first be selected. Identifying variables or factors (both dependent and independent) influencing the process is carried out in this phase, as is partitioning the data into training and test folds.
- Explore: In an exploratory stage, uni- or multivariate analysis is conducted to detect gaps in the data and to study interconnected relationships; this phase is expected to rely heavily on data visualization techniques.
- Modify: Data is cleaned and transformed in order to prepare it for the modeling, using insights from the previous exploration phase.
- Model: This phase is where the core modeling step applies, i.e., a variety of data mining techniques are applied with the intention of identifying the one most suitable for solving the business problem at hand.
- Assess: Evaluate the model (How useful and reliable is it? How well it solves the problem?). Computing quantitative evaluation metrics of the best model's quality is part of this last phase.

SEMMA has had limited impact to date, which is likely due to the fact that it was created (and is owned) by a single proprietary company.

3.3 CRISP-DM

In contrast to the KDD process, the Cross-Industry Standard Process for Data Mining (CRISP-DM for short) [18, 19] was developed in an industrial environment and emerged from the cooperation of the companies NCR System Engineering, SPSS Inc., and DaimlerChrysler AG. The iterative CRISP-DM starts with the Business Understanding phase before and ends with the Deployment phase after the cycle of the KDD process (Fig. 3). Because activities not previously considered are included here, the process is more comprehensive. The process is divided into the following six phases:

- Business Understanding: From the business perspective, project goals are determined and requirements and resources are defined in this initial phase. All findings are incorporated into a project plan.
- Data Understanding: Here, data is collected, described, and analyzed to explore it.
- Data Preparation: The goal of this phase is to create an adequate dataset for the following modeling. For this process, data is selected, cleaned, transformed, merged, and formatted.
- Modeling: In this phase, the modeling itself is performed. Therefore, modeling techniques are applied and parameters are calibrated.
- Evaluation: The obtained model is tested for its final use by evaluating how the defined business objectives from the Business Understanding phase have been achieved. In addition, past activities are reviewed, and next steps are determined.
- Deployment: The model and the knowledge gained through it are made usable.

During the use of the model, it is monitored and, if necessary, maintained. Besides that, the project is documented and a final report is prepared.

The recommended procedures of the CRISP-DM are described in a detailed handbook. It states that each phase has several generic activities, which in turn are associated with artifacts, mostly in the form of reports. A project that adapts the CRISP-DM in its pure form and is carried out entirely according to its manual will therefore result in a comprehensively documented project. However, the process is often adapted to the individual circumstances of a project, which leads to the omission of given elements or the addition of new ones [20]. It is still widely used today and can be considered the de facto standard in the field of data-intensive analytics projects [20–22].

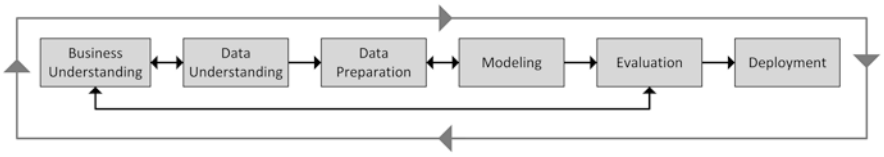

Fig. 3 The CRISP-DM methodology (simplified after Fig. 2 from [18, p. 10])

3.4 CRISP-ML(Q)

CRISP-ML(Q) (short for “Cross-Industry Standard Process for the development of Machine Learning applications with Quality assurance methodology”) is an attempt (by a group that did not include the original CRISP-DM creators) to adjust CRISP-DM from data mining to machine learning work [23].[3] It takes into account the special characteristics of machine learning, such as monitoring and maintaining a machine learning application in a changing deployed environment. As the name already indicates, essential concepts have been taken over from the CRISP-DM. In CRISP-DM(Q), the two phases Business Understanding and Data Understanding are merged into the Business and Data Understanding phase. The term “Maintenance” has found its way into the name of the Monitoring and Maintenance phase. Quality assurance measures to mitigate risks are proposed for all six phases of the iterative methodology.

3.5 Data-to-Value (D2V)

The Data-to-Value methodology (“D2V” for short) is a development process model [24–26] for the construction of systems that use (mostly supervised) machine learning in at least some of its components; it was developed, tested, and taught during the teaching of university students at various universities (Essex, Zurich, Frankfurt, Sheffield, Coburg) over the course of a decade, and it is motivated by the first author’s long-standing industry practice in research and development projects in natural language processing and information retrieval system construction of applications for professionals in the vertical domains of news/journalism, finance/insurance, legal, risk/security, and pharmacology. It offers several characteristics that are unique at the time of writing:

- It is an “evaluation first” methodology, which means that quantifying models is addressed before any models are actually built; evaluation scaffolding is constructed early, so ongoing quantitative evaluation can guide the development process, following the saying “what you can’t measure, you can’t improve.”
- It has an intricate number of stages – over 30 – each of which is associated with some from a set of 100+ guidance questions to help more junior team members around standard pitfalls and to create more consistency for senior team members.
- In particular, acknowledging the importance of data quality, the gold data annotation process is spelled out in detail, which is surprisingly lacking from most natural language processing and machine learning text books published to date.

[3] https://ml-ops.org/content/crisp-ml

- Ethical and technology impact considerations are not treated as an optional afterthought; following [27, 28], they have been integrated into the process by design, in the form of various checkpoints.
- D2V features a "feasibility study" phase early in a project, which was motivated by real-life requests by managers for impossible projects, in particular predictive modeling of target variables for which no predictive signal is available in the available datasets. This step dramatically de-risks data-intensive projects and helps reduce sunk cost.

Figures 4 and 5 show the various stages of the D2V methodology.

Note that the boxes with double-line frames are indicative of subprocesses. The feasibility study is a subgraph that contains a copy of the whole graph (except feasibility study itself) in a way that permits many steps to be partially completed or skipped; its purpose is, ahead of committing to a full-blown project, to check whether there is enough signal in the data so that the predictive models conceived to be constructed later actually can be built. Due to space reasons, we must refer the interested reader to the open access technical report [25], which contains the most detailed description of the process model to date.

Now that we have sketched the space of common and recent methodologies, we can look into the list of core requirements for a software system that supports these and other similar methodologies.

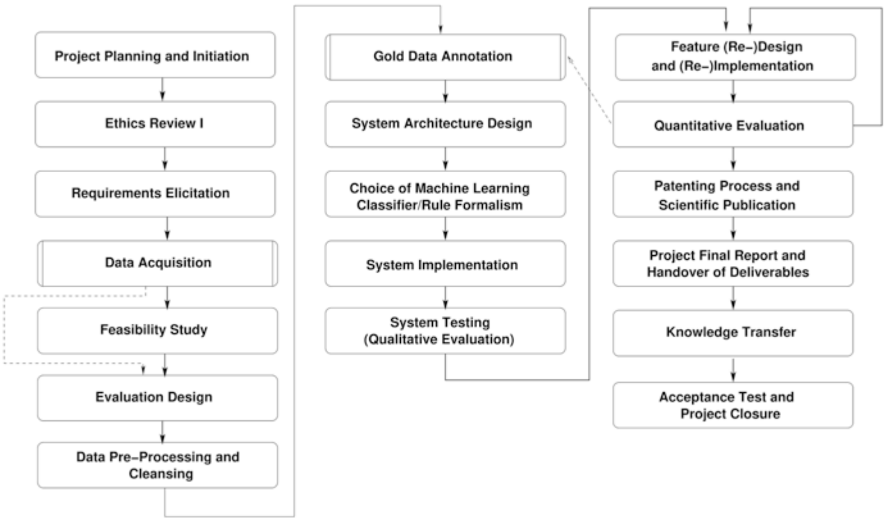
Project Planning and Initiation
Ethics Review I
Requirements Elicitation
Data Acquisition
Feasibility Study
Evaluation Design
Data Pre–Processing and Cleansing
Gold Data Annotation
System Architecture Design
Choice of Machine Learning Classifier/Rule Formalism
System Implementation
System Testing (Qualitative Evaluation)
Feature (Re–)Design and (Re–)Implementation
Quantitative Evaluation
Patenting Process and Scientific Publication
Project Final Report and Handover of Deliverables
Knowledge Transfer
Acceptance Test and Project Closure

Fig. 4 The Data-to-Value phases: overall process (after [26])

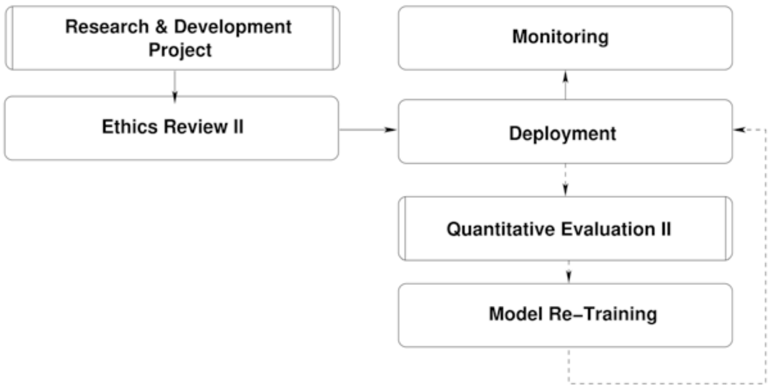

Fig. 5 Further Data-to-Value phases: (i) gold data annotation subprocess (left), (ii) system deployment (right; after [26])

4 Requirements for Software Support Tools

We sourced our requirements as follows: First, we derived many requirements directly or indirectly from properties of the methodologies that the software should support, especially where these overlap. Second, we supplemented these by requirements gained through introspection by extracting them from user stories by the two authors, who have a combined 30 years of experience.

We will describe our findings starting with user stories: A series of them put the users of the system (team, in particular PM, and other stakeholders) at the center. We then derive a longer list of requirements of instantiated requirement templates from these; for brevity, we show a selection of the most relevant requirements here, and we will exclusively focus on functional requirements. Finally, we derive a list of the key classes (in object-oriented thinking) or entities (in entity-relationship thinking), respectively.

4.1 Overall Vision

In addition to our presentation of the functional requirements, we begin with a vision statement to capture the overall spirit and scope of the system that we anticipate being a useful support tool for machine learning projects with the following particular methodologies:

There is a need for a system that guides a project team to follow a chosen machine learning methodology and which manages the project's metadata centrally, persistently, and transparently.

This vision of a system that supports its users by providing process guidance and metadata management is like a meta-level requirement: All other requirements should at least not be in conflict with it. By saying "chosen" methodology, we imply that not a particular methodology is hardwired in the software we envisage but that it should be designed and implemented in a way that is separate from any particular methodology and thus be capable of supporting most, past, present, and future methodologies; in order words, we attempt to generalize across several or most approaches. This requirement seems realistic, as in the realm of workflow tools, such a generic approach has already been successful (see below).

4.2 User Stories and Requirement Templates

We now present several user stories, followed by the requirements extracted from them.

Jack is an experienced project manager. His company uses a new software to support a range of machine learning methodologies, as it improves his everyday professional life in many ways: Instead of having to gather specifics for a machine learning project (phases, paths, quality, etc.) from multiple systems, he can now use one system one-stop shop tool to capture, store, and communicate a project's status and progress to his superiors and team members as well as to control and monitor his projects.

Requirement 1 (Track Current Phase) (Must-Have) The system shall maintain the current state (active phase) of the project with respect to the methodology that it uses.

Requirement 2 (Monitor and Control Project) (Must-Have) The system shall offer possibilities to monitor and control the project.

Requirement 3 (Track Actions and Time per Phase) (Must-Have) The system shall capture "what happens when" in the project, including the duration (in calendar days) spent in each phase.

Requirement 4 (Import Methodologies) (Must-Have) The system shall let the user model import one (from a predefined library of given methodology) workflow.

Requirement 5 (Edit/Customize Methodologies) (Must-Have) The system shall let the user customize a methodology's workflow for a project.

At the beginning of a project, Jack must define which methodology (CRISP-DM, Data-to-Value, etc.) should be used. The phases and paths are then automatically recorded in the software as a default workflow of the methodology. This workflow, including phases and paths, can be adopted or customized. Jack is able to enrich the selected workflow with project specifics, including the planned duration of sections, the project participants (core team) and other stakeholders, and their roles and the RACI (responsible, accountable, consulted, informed) category assigned to them [29]. All changes to the project and their history are automatically stored and versioned. Target times specified by him can be compared with the actual numbers, or with those of past projects. If guidance questions are provided by the methodology or requested by the users, then Jack can answer them himself or pass them on to his team, thereby determining common processes of the machine learning project. All answers are stored in the system's database.

Requirement 6 (Support Multiple Methodologies) (Must-Have) The system shall associate one methodology with each project.

Requirement 7 (Track Actions and Time per Phase) (Must-Have) The system shall log project actions conducted in a permanent read-only activity protocol.

Requirement 8 (Capture and Categorize Stakeholders) (Must-Have) The system shall capture all project stakeholder names, roles, and their associated RACI categories.

Requirement 9 (Create Artifacts) (Must-Have) The system shall automatically create document-based and/or document-like artifacts for evaluating completed phases and/or activities. (Note: An artifact is a by-product created during the machine learning development process, such as datasets, models, or documents.)

Requirement 10 (Ask Guidance Questions and Record Answers) (Must-Have) The system shall ask the project manager the set of potential guidance questions associated with a phase (if the chosen methodology posits them) and capture their answers. If they do not know an answer, permit entry of a list of team members ("share/forward") that may be able to answer.

Jack can enter needed skills and skill levels for the project and then adds Sarah to the project as a team member in a data science role. The system is aware of her skills in data science, Python programming, and in particular her experience with clustering methods; however, as the system detects that Sarah has not yet got experience with Apache Hadoop and Apache Spark, two standard systems for distributed processing in big data projects, which Jack had indicated as needed for the project, the system alerts Jack to the skill gap, and because the formal start of the project is still a few weeks away, Jack proposes Sarah be sent to a training course as a means of upskilling to her line manager Joe, a suggestion to which he agrees.

Requirement 11 (Manage Team) (Must-Have) The system shall capture the initial team, the role(s) of each member, skills required, skills needed for the project, as well as ongoing team changes during the project.

All project participants can view the status of the project and the progress in the workflow at any time on the central view of a web-based dashboard, while stakeholders external to the team get to see a specific view. The team can view their personal task inbox as well as the global task list for current phrase with respect to the methodology. Important success indicators, activities already carried out, and activities still to be carried out are shown. RAG tags (red-amber-green) are used to convey high-level status information to senior stakeholders. On the one hand, the visualization of the dashboards at the moment of viewing can serve as a basis for discussion vis-à-vis stakeholders. On the other hand, standard reports can be sent at regular intervals, or ad hoc reports can be sent via email or other communication channels when defined events occur. A final report is semiautomatically generated by the system at the end of each phase and at the end of each project. Jack can view the current project and also past or concurrent projects he is entitled to. These are stored in the system directly or through links to a project database. The presentation of several projects side-by-side enables them to be compared with each other, and all data is persisted for long-term archival purposes.

Requirement 12 (Manage Tasks) (Must-Have) The system shall permit team members to view their assigned (or all) tasks that are associated with the given methodology's current phase. They can also change the status, using the ternary states "OPEN," "IN PROGRESS," and "COMPLETED"; tasks are automatically synchronized with external third-party software systems from the project management and the information management domains.

Requirement 13 (Manage Scope Change) (Must-Have) The system shall permit the PM to enter a scope change request received from the customer, and it collects sign-offs and comments from relevant stakeholders.

Requirement 14 (Regularly Update Stakeholders) (Must-Have) The system shall send regular project updates to all stakeholders honoring the RACI matrix and comprising RAG status for all milestones, potential delays accrued, best model quality information, and key performance indicators.

Requirement 15 (Create Final Report) (Must-Have) The system shall automatically generate a final report after a project is completed. The report is stored in the system and may be edited by the project manager and/or technical lead before sending it to relevant stakeholders. The customer of the project is asked to acknowledge initial receipt, and by answering to what extent the project has met the criteria for success, the project is formally closed.

Maria has a computing degree specializing in data engineering. In the team, it is her task to import different input datasets and enrich them with metadata. The software system supports her in comparing and preparing different input datasets. She can add useful comments to each dataset for her team colleagues. The system supports her by offering the possibility to integrate several systems, which she uses as data sources.

Requirement 16 (Plug-Ins) (Must-Have) The system shall be able to integrate different types of software via a plug-in mechanism, so that a team/organization can choose from a broad range of external tools.

Requirement 17 (Track Input Datasets Versions) (Must-Have) The system shall capture and store a short description from a business perspective, attributes, and metadata of each input dataset for machine learning. (Note: Metadata is data about data, which is the description and context of data.)

Requirement 18 (Support Team Activities) (Nice-to-Have) The system should provide team members with the ability to support and comment on their activities.

Requirement 19 (Track Code Versions) (Must-Have) The system shall capture and version code the data used for machine learning experiments.

Requirement 20 (Record Decisions and Rationales) (Must-Have) The system shall capture (and store in a database) any important decisions taken and the rationales behind them.

Wendy, a data scientist on the project, has an idea for a new feature that may make her company's classification model more accurate. As she is in the Feature Engineering phase of Data-to-Value methodology used in her project, she can remain in the current phase and initiate a "new feature" hypothesis action. She then codes up the feature extractor for her idea and triggers the evaluation action. It seems to help, as F1 goes up slightly, priming her to checks in the new version of the code. The system keeps track of the rationale behind her idea and links it to the change-set ID of the revision control system that has the code, the model version thus improved, and the database of the system records the new quality scores (precision, recall, F1) of her improved classifier. The system also automatically updates work statistics, adding 1 day to the Feature Engineering phase. Two days later, at the end of the month, stakeholders receive an email to the system's dashboard that shows the quality improvements achieved in this month's reporting period.

Requirement 21 (Track Output Datasets Versions) (Must-Have) The system shall capture and version the success metrics, attributes, and metadata of each output record for machine learning. Note: Success metrics are, e.g., accuracy, error rate, precision, and recall.

Bob is a junior data scientist that has recently joined Wendy's team from a local university. He is smart and hard-working, but naturally, he still lacks real-life project experience. While, officially, Wendy mentors Bob, in practice she has to spend much time in meetings as she receives more and more responsibilities. Bob is guided by the system's phases, each of which is associated with guidance questions that help Bob avoid some typical "beginners' pitfalls." As he is charged with the data annotation subproject, the system suggests some possible tools for annotation and for inter-annotator agreement, and it persists the quality of each round of gold data annotation. After working hard in phase Annotate More Data by Multiple Annotators, Bob initiates a re-training action of the latest model based on additional gold data that was just annotated, and the system calls the training and evaluation scripts automatically. Upon completion of the automatic evaluation, quality KPIs are stored in the system's database with a time stamp. Seeing the new learning curve of a baseline support vector machine regressor, the system suggests checking if it has flattened enough to suggest concluding the gold data annotation work, and in doing so, he concurs. Bob spends the next couple of weeks training and improving various model variants, and he likes that the system provides a central means for him to keep track of the hyperparameter setting and the evaluation results of each model. He also draws motivation from seeing the F1 score improve over time in the "best model to date" view, which also shows the remaining time available for experimenting in the current project. In the past, Bob kept track of experiments in "README" files, but project managers would typically keep asking for them to be sent by email to them, which cluttered both their email inboxes.

Requirement 22 (Track Model Provenance) (Must-Have) The system shall capture and store the short name, version number (release number), technical description, hyperparameter settings, training data used, and revision control identifier (pull request ID) of each machine learning model. Ideally, this should be done to minimize human effort (avoiding duplication of data by semiautomatic import/software integration).

Requirement 23 (Track Quality over Time) (Must-Have) The system shall capture and store in a database the absolute quality as well as relative progress or regress of all model variants with time stamps.

Alice is the Chief Technology Officer (CTO), to whom the Vice President of Research and Development reports, and to whom directs Wendy's data science group. As a key stakeholder, she receives monthly PDF reports that document the project's progress, and she has access to the dashboard for the project. Skimming over the PDF report, it occurred to her to compare the project with the most similar past project to see how the current performance holds up to that scrutiny. Click on the dashboard; she can view the actions, milestones, and KPIs of the current project; and by adding a second project to this view and by using the built-in search, she can make the dashboard show the most similar project side-by-side for easy comparison. The fact that the system's database keeps track of all past projects' history and outcomes permits to use an organization's past performance in order to predict future behavior: Alice finds that the most similar past project was delivered successfully, with a small-time delay of 5%, and she is relieved that this may suggest that this time, absent of unknown unknowns, perhaps a similar result may materialize.

Requirement 24 (Track Artifacts) (Must-Have) The system shall store all artifacts versioned and with project assignment.

Requirement 25 (Search) (Must-Have) The system shall be able to search for artifacts, stakeholders, skills, and project knowledge.

We should stress that we do not assume our catalog to be complete or even complete with respect to "must-have" requirements, but merely as a first attempt to spell out the needs for a set of systems that – once implemented – can support a range of methodologies. We encourage others to supplement our list.

4.3 From Requirements Toward OO Classes/ER Entities

The above requirements help us identify a set of classes (or relational entities) as follows:

- *Projects.* The system shall maintain a persistent list of projects, some of which are in progress, some of which are completed, and some of which have been put on hold.
- *Team Members.* A team member has a line manager, a set of skills and associated availability information, and a set of roles in the project.
- *Tasks.* Tasks are the ultimate constituent elements of work packages in a project plan.
- *Roles.* A role is the function (or set of functions) that a team member plays in a given project, e.g., "technical lead," "project manager," "data scientist," "software engineer," or "data quality specialist."
- *Skills and Skill Levels.* A team member can have a particular skill profile (set of skills at certain levels each) associated with them. A project can have a set of skills and levels needed associated with it.
- *Stakeholders.* The PM can create and maintain a list of stakeholders (typically, people outside the project team), who may be external or internal and have an interest in the project. The project's executive sponsor, who pays for it, is a prime example. The system shall maintain RACI information (a classification of how to communicate with the stakeholder; see [29]). A stakeholder can pose a question to the PM. A stakeholder can view one of several progress dashboards.
- *Models.* A project relies on one or more machine learning components or models, each of which exists in various iterations. Models have a technical short name and a description, as well as link to code and training data it was used to induce it from. Models can be run on datasets (experiments), resulting in experimental results. Model performance can be plotted on a timeline that documents progress regarding quality (time *t* shows all models available then, singling out the best one).
- *Experiments.* A project typically comprises several experiments per machine learning component. An experiment comes into life once code for a model type is run against one or more folds of one or more datasets. Experiments can conclude once time allocated to experimenting is used up, sufficient quality has been reached, or (most often) diminishing returns on efforts invested have been reached.
- *Datasets.* A project uses one or more datasets. Datasets can be human-annotated or not ("raw"). Datasets can either play the role of input or source of gold data for either training or evaluation.
- *Methodologies*. A methodology defines a set of phases and possible transitions between them in the form of a directed graph. Each phase may contain activity information and other elements such as questionnaires.
- *Textual Artifacts.* A textual artifact is a text document (regardless of file format) that is uploaded (mostly automatically) to serve as the documentation for the

project as a whole or serves as the result (deliverable) of a methodology's phase. One can view these as file attachments.
- *Phases.* A phase is a state (point in time associated with a set of activities and project subgoals) in a methodology; it can recursively contain other phases.
- *Activities.* An activity is something that a team member is expected to carry out, given the project is in a certain stage of a methodology.
- *Questionnaires.* A questionnaire is a set of predefined questions associated with some methodologies and their associated, project-specific answers. They support making assumptions and decisions in the project explicit, transparent, and shareable.
- *Code.* Model implementation program code is not contained in the system but referred to as a link to a source code change-set (e.g., in Git or a similar revision control system).

5 Related Work

We found five broad groups of prior art that are relevant to the work presented in this chapter: first, work on tool support for machine learning methodologies; second, relevant work on requirements capture, including requirement specifications for machine learning models; third, tools to construct machine learning pipelines; fourth, general work on tooling support for workflows, especially but not limited to the software engineering domain; and fifth, work on tool support for machine learning development, deployment, and operations.

5.1 Work on Tooling for Machine Learning Process Methodology

A useful starting point for abstracting over individual methodologies is comparisons: For instance, several papers offer survey and comparisons of CRISP-DM, SEMMA, KDD, and other methodologies [22, 30–34].

In a case study [35], 17 software engineers, data scientists, and others at a Dutch bank were interviewed to identify shortcoming of existing machine learning process models. They find that "existing development tools for machine learning are still not meeting the particularities of this field," and in particular, "feasibility assessments, documentation, model risk assessment, and model monitoring are stages that have been overlooked by existing lifecycle models."[4]

We are not aware of any previous work on actual requirements elicitation for software support tooling in the context of following a methodology.

[4] The interviews predate the major Data-to-Value publications, which address several aspects of all of these.

5.2 Requirements Capture

The computer science literature on requirements capture for designing software systems is vast, and it is covered in general software engineering textbooks [10, 11] as well as in dedicated monographs and papers [4, 12, 36–41], so we will just mention a few exemplary references important to our work here.

Mullery [42] describes an early method for controlled requirement specification. User stories have long been considered a useful tool in requirements elicitation [43].

Recently, design thinking has had a great influence on software engineering, in particular under the various agile methodologies. Canedo and Parente da Costa [44] provide a survey of the intersection of design thinking and agile software engineering. While we are not aware of previous work that derived requirement templates from user stories, in [45] goals are derived from a use-case based requirement specification. While we extracted the requirements from user stories manually, [46] propose an interesting idea, namely, the extraction using natural language processing methods (see [47] for a review).

5.3 Workbenches for Constructing Machine Learning Pipelines

There are a number of software systems that permit the technical experimentation with machine learning methods by defining workflows run by individuals, and without addressing any nontechnical issues such as project management, data and code provenance, etc. We will just name two popular and free example systems here, but there are many others. Weka (short for: Waikato Environment for Knowledge Analysis, [48]) is an open source project originating from the University of New Zealand at Waikato. It is implemented in Java, therefore cross-platform, and it lets users define data mining/machine learning workflows visually. These can then be executed end to end automatically at the click of a button from reading the data to showing an evaluation result table.[5] Orange [49] is a similar offering originating at the Bioinformatics Laboratory, Faculty of Computer and Information Science, University of Ljubljana, Slovenia. It is implemented in Python and C++ and likewise uses a graphical "no code" interface to define local data processing workflows that constitute machine learning experiments.[6] RapidMiner (formerly known as YALE) is a commercial offering originating from work done at the University of

[5] See http://old-www.cms.waikato.ac.nz/~ml/weka/ (accessed 2023-01-30) and https://en.wikipedia.org/wiki/Weka_(machine_learning) (accessed 2023-01-30).

[6] See https://orangedatamining.com/widget-catalog/ (accessed 2023-01-30) for a set of the available processing resources/components that can be used in a workflow and https://www.youtube.com/c/OrangeDataMining (accessed 2023-01-30) for a set of training videos.

Dortmund, Germany.[7] It is cross-platform and also permits the composition of data flows on screen via visual programming. SAS Inc., the largest privately owned software company, also has a range of offerings in this space (SAS Enterprise Miner, SAS Viya, etc.).[8]

These and similar systems are very useful for beginners or machine learning users without the skills or willingness to implement code, but contrary to our goal here they do not integrate documentation, team collaboration, and other nontechnical aspects that are essential for working in large and in particular distributed teams.

5.4 Work on Support Tooling for Business Workflows

There is a huge body of work in computer science and business informatics specifically on workflows[9] and computer-supported systems for workflow management (see [50] for a survey).

Besides the academic literature, there are software products available for BPM (business process modeling) and RPA ("robotic process automation"), in particular from the largest enterprise software providers: SAP SE's Signavio Process Manager and Oracle's BPM Suite.[10] The latter contains Oracle BPM Studio, a component that enables process developers to create process-based applications and for process analysts and developers to model business processes.

5.5 Work on Support Tooling for Machine Learning Development, Deployment, and Operations

The term "ML-DevOps" combines the areas of development, deployment, and operations of machine learning applications (in analogy to traditional software DevOps, cf. [51]) so that their cooperation is strengthened with a continuous pipeline. The objective is to reduce the release time of usable software products and to increase the quality of the software product in the production environment [52, 53].

[7] See https://rapidminer.com (accessed 2023-01-30).

[8] See https://www.sas.com (accessed 2023-01-30).

[9] In computer science, "workflow" is a highly ambiguous term, which may denote business processes (the word sense we are interested in here), allocation of work to workers in operating systems or high-performance compute clusters, and global distribution of work in grid computing (distributed scientific computing), among others.

[10] See https://www.oracle.com (accessed 2023-01-30) and https://www.sap.com (accessed 2023-01-30).

Available software products in this area are Domino's Enterprise MLOps platform, DataRobot's AI Cloud platform, and Weights & Biases's MLOps platform.[11] The differences to the software proposed in this chapter are the lack of integration of (and guidance by) different (customizable) methodologies and the lack of team functionality. This results in disadvantages: For example, a question catalog with guidance questions can be provided by a methodology to help in going through the phases. This question catalog could recommend certain workflows and exclude other workflows based on the response behavior of the team members.

6 Discussion

We have collected what we believe are the core requirements for a support tool that facilitates implementing (following, complying with) recently proposed machine learning methodologies, and we have shown a subset of higher-level ones in this chapter (some more granular ones were not presented for space reasons). The main lesson to be learned from this exercise is a reaffirmation that considerable value can be created through the combination of well-designed processes supported by well-designed support tooling; however, a strong symmetry can be observed in that the advantages benefit more the organization and management than the team members. All team members need to embrace the methodology as well as the tool, which requires a small overhead of time and effort from everyone; however, the immediate benefit is to have a single go-to location where project information can be accessed from. Project managers and less experienced team members benefit the most from the tooling as specified here, the former from the centralization of project-related information and the latter from the guidance that the tooling provides.

It is hard to check a set of requirements for completeness in isolation; the easiest way to find out that everything that is needed is covered is perhaps to implement the requirements in a running system, so that shortcomings will quickly become apparent. Ultimately, support tooling for methodologies ought to be evaluated in controlled experiments, as has been proposed for methodologies themselves [54], but the effort to carry out such experiments is substantial.

One of the hardest problems, in our view, will be to design any supporting software in ways that ensure it will be embraced or at least accepted by all team members. The past has shown that many systems (including useful and well-respected ones like Atlassian's JIRA issue tracking system) are rejected as "bureaucratic" or "overkill," whereas others (e.g., version control systems like Git and the online service offering it, Microsoft's github.com, or the team chat groupware Slack) have become more readily accepted. One key to success here might be offering

[11] See https://www.dominodatalab.com/product/domino-enterprise-mlops-platform, https://www.datarobot.com/platform/ and https://wandb.ai/site (accessed 2023-01-30).

alternative user interfaces (e.g., based on the command line) to accommodate varying preferences among software engineers.

Another serious practical challenge is the question of how to design the software tool that supports the methodology in a way so that the integration of existing software tools is possible from the project management domain (e.g., Microsoft Project, Omni Group's OmniPlan, Atlassian Trello, Atlassian JIRA, SAP Project System, etc.), the machine learning experimentation domain (e.g., RapidMiner, Python/Google Tensorflow, Weka, JetBrains PyCharm, JetBrains IntelliJ, Microsoft Code), team and communication domain (e.g., email, Slack, Zoom), and data and information management domain (e.g., PostgreSQL, Git, Amazon Web Services Simple Storage Service) (Fig. 6). Project plans need to be importable, and activities belonging to a methodology's phases must be able to trigger creation of the tasks assigned to team members, for instance, in JIRA, automatically.

The following four refined yet minimal requirements fall out of the above discussion and can be seen to specialize the original Requirement 16. They appear to be critical for the system's practical success. At the same time, it is challenging to implement them in a way that avoids the anticipated system to become more than loosely coupled with the various third-party systems supported. The plug-in mechanism, therefore, must be designed to be simple, generic, portable, and minimalistic.

Requirement 26 (Import Tasks) (Nice-to-Have) The system should provide an inbound interface for third-party plug-ins to import sets of tasks from external project management systems (e.g., Microsoft Project, Omni Group's OmniPlan).

Requirement 27 (Export Tasks) (Nice-to-Have) The system should provide an outbound interface for third-party plug-ins to export sets of tasks to external ticketing management systems (e.g., Atlassian JIRA or Trello).

Requirement 28 (Synchronize Contacts, Skills, and Comments) (Nice-to-Have) The system should provide an inbound and outbound interface for third-party plug-ins to import and export project-related communications (e.g., email, Slack), people names, roles, and contacts (e.g., Lightweight Directory Access

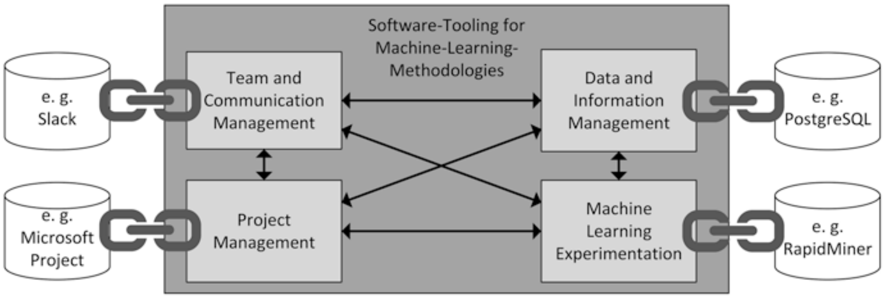
Software-Tooling for Machine-Learning-Methodologies
Team and Communication Management
Data and Information Management
Project Management
Machine Learning Experimentation
e. g. Slack
e. g. Microsoft Project
e. g. PostgreSQL
e. g. RapidMiner

Fig. 6 Loosely coupled integration of external systems

Protocol, Internet Message Access Protocol, Microsoft Exc[]hange) from and to external communication as well as people skills, expertise, and experience via personnel management systems (e.g., SAP SuccessFactors, Kenjo, Workday).

Requirement 29 (Synchronize Development Activity) (Nice-to-Have) The system should provide an inbound and outbound interface for third-party plug-ins to import and export action or change notifications from and to external experimentation management systems (e.g., SAS, RapidMiner) and data (incl. model) stores (e.g., RNA Mapping Database, Amazon Web Services Simple Storage Service).

7 Summary, Conclusions, and Future Work

In this chapter, we have described a set of requirements for software tooling support for machine learning methodologies. We tried to do so without hard-wiring assumptions of any specific methodology, abstracting the narration to the level of following a predefined workflow.

Our result is a list of 29 presented functional requirements, with indications whether we rate each as essential or optional. In drawing up this list, we followed the proven template-based approach for requirements capture.

While there has been a lot of prior research on business process modeling, including software tooling, we are not aware of any previous requirements capture attempt for machine learning methodology software support.

In future work, our findings could be confirmed or challenged by holding interviews with experienced stakeholders to ensure our catalog of requirements is complete. High-level user requirements ought to be translated to specific system requirements that in turn inform a system architecture for the envisaged system. A system or a set of systems then ought to be implemented that embody subsets of these requirements as the logical next step. A prototype could then serve to affirm the consistency and completeness of the requirements.

While this is going to be a lot of work, it might not be hard work; in contrast, the adoption of such tools ought to be maximized, and the effectiveness of the support provided by such tools ought to be evaluated (e.g., using questionnaires similar to Lending and Chervany [55]), tasks we consider both very challenging. Furthermore, future work could study how to maximize adoption of tools by the various individuals that make up project teams and other stakeholders.

Acknowledgments The authors would like to thank Marco Zierl for discussions and two anonymous referees for helpful feedback that improved the quality of this chapter.

References

1. Weber, C., Hirmer, P.: P. Reimann. In: Abramowicz, W., Klein, G. (eds.) Business Information Systems, pp. 403–417. Springer International Publishing, Cham (2020)
2. Iivari, J.: Commun. ACM. **39**(10), 94 (1996)
3. Cheng, B.H., Atlee, J.M.: Future of Software Engineering (FOSE '07), pp. 285–303 (2007). https://doi.org/10.1109/FOSE.2007.17
4. Kotonya, G., Sommerville, I.: Requirements Engineering. Worldwide Series in Computer Science. Wiley, Nashville (1998)
5. Sommerville, I.: IEEE Softw. **22**(1), 16 (2005). https://doi.org/10.1109/MS.2005.13
6. Kreuzberger, D., Kühl, N., Hirschl, S.: Machine learning operations (mlops): overview, definition, and architecture (2022). https://doi.org/10.48550/ARXIV.2205.02302. https://arxiv.org/abs/2205.02302
7. Provost, F., Fawcett, T.: Data Science for Business. O'Reilly Media, Sebastopol (2013)
8. Taulli, T.: Implementing AI Systems. Apress, Berkeley (2021)
9. IEEE: IEEE standard glossary of software engineering terminology (1990). https://doi.org/10.1109/IEEESTD.1990.101064. IEEE Std 610.12-1990
10. Sommerville, I.: Software Engineering, 10th edn. Pearson Education, London (2015)
11. Braude, E.J., Bernstein, M.E.: Software Engineering, 2nd edn. Wiley, Nashville (2007)
12. Hull, E., Jackson, K., Dick, J.: Requirements Engineering. Springer-Verlag, London (2005)
13. The SOPHISTs: Requirements engineering: the sophists »a short RE primer« (2016). https://www.sophist.de/fileadmin/user_upload/Bilder_zu_Seiten/Publikationen/Wissen_for_free/RE-Broschuere_Englisch_-_Online.pdf. Accessed 2022-07-21
14. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: AI Mag. **17**, 3 (1996). https://doi.org/10.1609/aimag.v17i3.1230
15. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: Commun. ACM. **39**(11), 27 (1996). https://doi.org/10.1145/240455.240464
16. Fayyad, U.M., Piatetsky-Shapiro, G., P.: Smyth. In: Fayyad, U.M., et al. (eds.) Advances in Knowledge Discovery and Data Mining. MIT Press, Cambridge, MA (1996)
17. SAS Institute Inc.: Data mining using SAS Enterprise Miner: a case study approach (2013). https://support.sas.com/documentation/onlinedoc/miner/ casestudy_59123.pdf. Accessed 2022-07-21
18. Chapman, P., et al.: CRISP-DM 1.0 – step-by-step data mining guide. Tech. rep. The CRISP-DM Consortium (2000) https://www.kde.cs.uni-kassel.de/wp-content/uploads/lehre/ws2012-13/kdd/files/CRISPWP-0800.pdf. Accessed 2008-05-01
19. Shearer, C.: J. Data Warehous. **5**, 13–22 (2000)
20. Schröer, C., Kruse, F., Gómez, J.M.: Proc. Comput. Sci. **181**, 526 (2021). https://doi.org/10.1016/j.procs.2021.01.199
21. KDnuggets: What main methodology are you using for your analytics, data mining, or data science projects? poll (2014). https://www.kdnuggets.com/polls/2014/ analytics-data-mining-data-science-methodology.html. Accessed 2022-07-21
22. Saltz, J., Hotz, N.: 2020 IEEE International Conference on Big Data (Big Data), p. 2038–2042 (2020). https://doi.org/10.1109/BigData50022.2020.9377813
23. Studer, S., Bui, T.B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., Müller, K.R.: Machine Learning and Knowledge Extraction. **3**(2), 392 (2021). https://doi.org/10.3390/make3020020
24. Leidner, J.L.: Project management for data science. Tutorial held at the IEEE international conference on data science and applications (DSAA 2018), Turin, Italy, 2018
25. Leidner, J.L.: Data to value: an 'evaluation-first' methodology for natural language projects. Tech. rep. Cornell University, New York, NY, USA (2022) https://arxiv.org/abs/2201.07725. ArXiv Pre-Print Server
26. Leidner, J.L.: In: Rosso, P., Basile, V., Martínez, R., Métais, E., Meziane, F. (eds.) Natural Language Processing and Information Systems: proceedings of the 27th International

Conference on Applications of Natural Language to Information Systems, 15–17 June, Valencia, Spain, pp. 517–523. Springer, Cham, Switzerland, NLDB 2022 (2022). https://doi.org/10.1007/978-3-031-08473-7_48

27. Hovy, D., Spruit, S.L.: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, pp. 591–598. ACL, Berlin, Germany (2016)
28. Leidner, J.L., Plachouras, V.: Proceedings of the First ACL Workshop on Ethics in Natural Language Processing Held at EACL, pp. 30–40. Association for Computational Linguistics, Valencia, Spain (2017). https://doi.org/10.18653/v1/W17-1604
29. Kerzner, H.: Project Management: a Systems Approach to Planning, Scheduling, and Controlling, 10th edn. Wiley, Hoboken (2009)
30. Azevedo, A., Santos, M.F.: Proceedings of the IADIS European Conference on Data Mining, 24–26 July 2000, pp. 182–185, Amsterdam (2008)
31. Haertel, C., Pohl, M., Nahhas, A., Staegemann, D., Turowski, K.: PACIS 2022 Proceedings (2022). https://aisel.aisnet.org/pacis2022/242
32. Kurgan, L.A., Musilek, P.: Knowl. Eng. Rev. **21**(1), 1 (2006). https://doi.org/10.1017/S0269888906000737
33. Mariscal, G., Óscar Marbán, C., Fernández: Knowl. Eng. Rev. **25**(2), 137 (2010). https://doi.org/10.1017/S0269888910000032
34. Martinez, I., Viles, E., Olaizola, I.G.: Big Data Res. **24**, 100183 (2021). https://doi.org/10.1016/j.bdr.2020.100183
35. Haakman, M., Cruz, L., Huigens, H., van Deursen, A.: Emp. Softw. Eng. **26**(5), 1 (2021)
36. Wiegers, K.: Software Requirements, 3rd edn. Microsoft Press, Redmond (2013)
37. Pohl, K.: Requirements Engineering: Grundlagen, Prinzipien, Techniken, 2nd edn. dpunkt, Heidelberg, Germany (2008)
38. Vessey, I., Conger, S.A.: Commun. ACM. **37**(5), 102 (1994). https://doi.org/10.1145/175290.175305
39. Herzwurm, G., Schockert, S., Mellis, W.: Joint Requirements Engineering: QFD for Rapid Customer-Focused Software and Internet-Development. Vieweg, Wiesbaden (2000)
40. Rupp, C.: Requirements-Engineering und -Management. Professionelle, iterative Anforderungsnanalyse für IT-Systeme. Hanser, Munich, Germany (2001)
41. Firesmith, D.: Engineering security requirements. J. Object Technol. **2**(1), 53–68 (2003)
42. Mullery, G.P.: Core -a method for controlled requirement specification (1979)
43. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E., Brinkkemper, S.: International Working Conference on Requirements Engineering: foundation for Software Quality, pp. 205–222. Springer (2016)
44. Canedo, E.D., da Costa, R.P.: In: Marcus, A., Wang, W. (eds.) Design, User Experience, and Usability: theory and Practice, pp. 642–657. Springer International Publishing, Cham, Switzerland (2018)
45. Anton, A.I., Carter, R.A., Dagnino, A., Dempster, J.H., Siege, D.F.: Requir. Eng. **6**, 62 (2001)
46. Ghosh, S., Elenius, D., Li, W., Lincoln, P., Shankar, N., Steiner, W.: ARSENAL: automatic requirements specification extraction from natural language. Tech. rep. Cornell University, New York, NY, USA (2016)
47. Raharjana, I.K., Siahaan, D., Fatichah, C.: IEEE Access. **9**, 53811 (2021). https://doi.org/10.1109/ACCESS.2021.3070606
48. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Data Mining: practical Machine Learning Tools and Techniques, 4th edn. Morgan Kaufmann, Amsterdam (2016)
49. Demšar, J., Curk, T., Erjavec, A., Črt Gorup, T., Hočevar, M., Milutinovič, M., Možina, M., Polajnar, M., Toplak, A., Starič, M., Štajdohar, L., Umek, L., Žagar, J., Žbontar, M., Žitnik, B.Z.: J. Mach. Learn. Res. **14**, 2349 (2013)
50. La Rosa, M., Aalst, W.M.P.V.D., Dumas, M., Milani, F.P.: ACM Comput. Surv. **50**(1) (2017). https://doi.org/10.1145/3041957
51. Ebert, C., Gallardo, G., Hernantes, J., Serrano, N.: IEEE Softw. **33**(3), 94 (2016)

52. Bass, L., Weber, I., Zhu, L.: DevOps: a Software Architect's Perspective. Addison-Wesley Professional (2015)
53. Lwakatare, L.E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M., Lassenius, C.: Inf. Softw. Technol. **114**, 217 (2019). https://doi.org/10.1016/j.infsof.2019.06.010
54. Saltz, J., Shamshurin, I., Crowston, K.: Proceedings of the Hawaii International Conference on System Sciences, pp. 1013–1022. HICSS (2017)
55. Lending, D., Chervany, N.L.: Proceedings of the 1998 ACM SIGCPR Conference on Computer Personnel Research, pp. 49–58. ACM, New York, NY, USA (1998)