Software and Systems Modeling (2022) 21:1389–1410
https://doi.org/10.1007/s10270-021-00926-6

SPECIAL SECTION PAPER

Holistic data-driven requirements elicitation in the big data era

Aron Henriksson[1] · **Jelena Zdravkovic**[1]

Received: 5 April 2021 / Revised: 8 September 2021 / Accepted: 13 September 2021 / Published online: 12 October 2021

Abstract
Digital transformation stimulates continuous generation of large amounts of digital data, both in organizations and in society at large. As a consequence, there have been growing efforts in the Requirements Engineering community to consider digital data as sources for requirements acquisition, in addition to human stakeholders. The volume, velocity and variety of the data make requirements discovery increasingly dynamic, but also unstructured and complex, which current elicitation methods are unable to consider and manage in a systematic and efficient manner. We propose a framework, in the form of a conceptual metamodel and a method, for continuous and automated acquisition, analysis and aggregation of heterogeneous digital sources that aims to support data-driven requirements elicitation and management. The usability of the framework is partially validated by an in-depth case study from the business sector of video game development.

Keywords Data-driven requirements engineering · Big data · Requirements modeling · Machine learning · Natural language processing

1 Introduction

Requirements are traditionally elicited through interactions with system stakeholders, where the relevant information is obtained mainly through interviews [1, 2]. Owing to the ongoing digital transformation across many industries, as well as the proliferation of social media, it has become highly relevant to consider digital data sources for elicitation of system requirements, in addition to human stakeholders. In the requirements engineering (RE) community, there are increasing efforts to support and enrich requirements elicitation by automatically collecting and processing digital data as additional sources for requirements acquisition [3, 4]. There are a wide range of digital sources that may be exploited, but recently increasing attention has been given to sources that are more *dynamic*, i.e., continuously generating large amounts of data through various mediums and platforms, such as online forums and microblogs, but also machine-generated logs and sensor data. The main motivation behind data-driven requirements elicitation is to exploit the availability of large amounts of digital data and thereby (i) consider a wider scope of sources of requirements, including potentially large and dispersed user bases, (ii) enable increased automation in processing relevant data by leveraging AI techniques based on, for instance, natural language processing and machine learning [6], (iii) facilitate continuous elicitation by rapidly considering the emergence of newly generated data, which helps to support software evolution through more frequent releases. Although the potential of digital data as sources of information for system requirements is becoming well-recognized, there are numerous challenges in processing this data effectively such that it can be fully exploited for development and evolution of enterprise software [7].

In contrast to conventional elicitation, where the requirements are elaborately elicited and guided by requirements engineers in stakeholder interviews, digital data sources are often non-intended, i.e., the information is not originally created for purposes of requirements elicitation and therefore is limited in terms of completeness, may be ambiguous, and lack structure. Being either human-generated, in the form of unstructured natural language, or machine generated, in the form of sensor data or computer logs, the data moreover tends to provide only implicit feedback on requirements; it is there-

Communicated by Dominik Bork and Janis Grabis.

✉ Aron Henriksson
aronhen@dsv.su.se

Jelena Zdravkovic
jelenaz@dsv.su.se
https://www.su.se/profiles/jzdra

[1] Department of Computer and Systems Sciences, Stockholm University, Postbox 7003, 164 07 Kista, Sweden

fore an inherent and fundamental challenge to transform the obtained raw data toward some canonical requirement format that is understandable to a software team and feasible to develop and implement, such as to the structured requirement template in the plan-based approach [2], or to the user story template in agile approaches [8]. Consequently, data-driven elicitation outcomes risk to be of low quality and practical use [8], or requiring a substantial manual effort [7, 9]. Furthermore, the structure and semantics of the data obtained from different sources vary significantly, leading to numerous challenges concerning how to process and aggregate data into coherent requirements. This is a highly complex task, and because business organizations are increasingly interested in exploiting the value of digital data sources for RE, they face the challenge of understanding how to apply data-driven RE. For many organizations, potentially relevant digital data sources generate massive amounts of data that come in many different shapes and forms, which adds significant complexity to data collection, storage and, in particular, analysis.

Our research aims to address this problem by proposing a structured and holistic approach to data-driven requirements elicitation. The relevant research questions are:

- *RQ1* How can different digital data sources, and different means of processing these, be conceptually modeled and related for eliciting system requirements?
- *RQ2* How can the conceptual model and the process for its use be validated to demonstrate the feasibility of the proposed framework?

In relation to RQ1, we have proposed a model-based approach for structuring and relating the concepts of different digital data sources, the means for their processing, and the mapping to system requirements; and for using it as a design basis for developing automated acquisition, structuring, and aggregation of heterogeneous digital data, toward mapping to requirements artifacts [5]. In this study, we address RQ2 by (i) elaborating the elicitation process, (ii) carrying out a partial empirical validation of the metamodel's correctness and usability through a case study using real data from a large gaming company, and (iii) by developing a prototype for analyzing the case study data using machine learning, in particular for classification and sentiment analysis, and comparing the distribution of predictions for three different games.

The proposed framework is intended to complement existing stakeholder-driven approaches to requirements elicitation by setting up a conceptual basis and a process for linking heterogenous digital data and means for their processing to requirements artifacts through—as far as possible—automation. While we envision that it is possible to automate most, but perhaps not all, of the activities in the process, the validation of the proposed framework in this study is partial and does not include all possible types of data and, moreover, constitutes a combination of automated and manual analyses. Manual analysis is performed in order to allow for illustration of the entire process; however, further research is needed for investigating how best to aggregate and map candidate requirements to requirements artifacts, as well as how best to exploit machine-generated data in the form of computer logs and sensor data.

The rest of the paper is organized in the following way. Section 2 provides a brief background to the relevant approaches for requirements elicitation, and an overview of previous research concerning requirements elicitation from digital sources. Section 3 presents the holistic metamodel for processing of digital sources and linking to requirements artifacts, and a process for its use. Section 4 seeks to validate parts of the theoretical proposal through an in-depth case study involving a global video game development company. A discussion and concluding remarks are presented in Sect. 5 and 6, respectively.

2 Background

This section provides a brief background to the main approaches to requirements elicitation, as well as overview of previous work in data-driven RE, focusing primarily on requirements elicitation from digital data sources.

2.1 Requirements elicitation

The main activity in RE concerns elicitation of information from available sources, which are primarily human stakeholders. Traditionally, the main goal of different elicitation techniques, such as interviews, workshops and observations, is to effectively support the requirements engineer in obtaining relevant information from stakeholders [2]. The collected requirements are specified in a structured way using a notation, i.e., template, for fostering a canonical syntactic structure.

The plan-driven process emphasizes the use and combination of different elicitation techniques, thus facilitating a rich elicitation that is then documented in detail, including main requirements types: functional requirements—tasks to be performed by the system—and quality requirements for the system behavior—some of which are also referred to as "constraints" in the literature. In many projects, requirements elicitation starts from goals, as they describe the intentions of stakeholders with respect to how the system should support business aspects. Through brainstorming and decomposition—in a process driven by the requirements engineer—the goals are further mapped to functional or quality requirements. While functional requirements have the "clear cut" determination criterion, quality requirements are

often elicited as goals and need to be further elaborated to become measurable.

Agile requirements elicitation also relies on collecting information from various human stakeholders in parallel, i.e., during interactive sessions between the agile team and stakeholders, the requirements are sketched as user stories [8]. The requirements are sometimes collected as "epics," i.e., large user stories resembling rather goals than concrete tasks. All of them are groomed in the product backlog to a suitable size to be developed in an iteration according to a given priority. In addition to desired functionalities represented by user stories, the backlog contains quality requirements, e.g., performance and reliability [2]. The product owner is the main authority responsible for deciding which requirements are of interest for development and the order in which to implement them.

Even though the two approaches differ in terms of working practices and roles, as well as in the templates used for requirements, they share some common aims, such as the need for continuous [10] and extensive requirements elicitation (which in plan-based approaches is done more upfront; in agile as late as possible), classification of requirements types, and prioritization. Recent studies have, in fact, reported the empirical perspective that the two approaches are often mixed in practice, especially in large-scale agile development [11] and that it is possible to combine them [12].

2.2 Data-driven requirements elicitation

Many studies have lately been addressing the need to elicit requirements from digital sources. One research track has focused on eliciting requirements from static data sources, typically domain knowledge [13], e.g., business documents [14], various types of models [15], or from software repositories, e.g., in order to support the elicitation of quality requirements [16]; the elicitation of data-driven quality requirements was also the focus of another study [17], emphasizing that these are typically collected as goals and need further refinement toward tangible requirements requests. Another, more recent research track concerns requirements elicitation from more dynamic, online data sources, motivated by the increasing inflow of user feedback. Common sources are discussion forums [18], online and app reviews [19], microblogs, e.g., Twitter [20], and mailing lists [21]. While this digital data is created by humans and primarily in the form of natural language, there are also research studies mining requirements from machine-generated sources, such as usage data [22] and sensor data [23]. Most of these efforts have focused on identifying and classifying requirements-related information from a single source; however, some studies have discussed techniques for combining user feedback and machine-generated data [24, 25]. A systematic literature review of data-driven requirements elicitation techniques showed that more research is needed to develop methods for eliciting requirements from machine-generated data and also that there is still a lack of methods that combine more diverse types of data sources [26].

Recently, some holistic views on data-driven requirements elicitation have been presented: [3] proposed an approach for integrating information originating from different IS using a domain ontology to serve as the baseline for the data model. [27] is similar in the sense that it relies on a domain ontology, with the addition of a high-level proposal for storing the final outcome of the elicited requirements originating into a connected graph. Previous efforts differ from ours as they do not focus on external data sources, nor do they encompass algorithms for data processing. The study presented in [28] defines a metamodel for method chunks built upon activities, artifacts, roles, and tools used across the entire RE process; as such, it differs from our approach, where metamodeling has been used to define and conceptually relate the integration of heterogeneous data sources, their processing, and mapping processed data to requirements.

When the data is mainly in the form of free text, natural language processing (NLP) is applied in order to extract requirements information. Identification of information in free-text is typically achieved through named entity recognition (NER), which enables classification of sequences of words [29] and identifies the beginning and end of any type of predefined requirements-related information, e.g., mentions of a functionality. Sentiment analysis is an NLP task applied to enable classification of a type of emotion or attitude, typically as positive, negative, or neutral [30]. Classification is the task of automatically predicting one or more classes that a set of data belongs, e.g., whether a review contains a feature request or a bug report. NLP increasingly makes use of supervised machine learning (ML), which relies on access to labeled data—often manually annotated to creating training data. This can be costly, but once an accurate and complete ML model has been created, it can be applied repeatedly to facilitate automation of requirements elicitation.

3 Integration of digital data to requirements

This section presents the theoretical foundation for the conceptualization and the process for analysis and aggregation of big data for requirements elicitation. The presented results are based on a previous publication [5], and in this study they are elaborated and improved based on a prototype implementation and an in-depth case study using real data from the business sector of game development.

3.1 Metamodel for data-driven requirements elicitation

One of the key challenges concerns the variety of big data, i.e., the fact that the data comes in many different shapes and forms, which adds significant complexity to data collection, storage and, in particular, analysis. The metamodel in Fig. 1 is aimed to define heterogeneous digital data sources, the means for their processing, and mapping of automatically analyzed data to requirements artifacts. The main purpose of the metamodel is to provide a model basis for structuring and designing the process of requirements elicitation from big data.

The metamodel was developed by the authors, and then assessed for correctness, completeness and usability by a senior academic, a data architect, and a Scrum developer [5]; in the process of this study, the metamodel was also reviewed by the gaming company (Sect. 4.6). The main differences from [5] include (i) refinement of some attributes as the result of the empirical validation presented in Sect. 4, (ii) generalization of the part related to requirements in terms of elicitation approaches.

The metamodel distinguishes: a classification of digital data sources (in yellow), the elements for processing the data (in gray), the elements for data aggregation and further mapping to requirements (in light olive), and a conceptualization of the requirements artifact and related elements (in white). The depicted classes and relationships are in detail described in Table 1.

3.2 Requirements elicitation process

The process of collecting digital data and mapping the data to requirements based on the conceptualization defined in the previous section is illustrated in Fig. 1. The main four activities in the process are *data collection* from different digital sources, *analysis* by different means of data processing, *aggregation* of analyzed data from one or more sources into candidate requirements, and *mapping* of the candidate requirements to requirements artifacts—new or existing. The inputs to data collection are various, potentially heterogeneous sources of data, generated by humans and machines. The output of the process may be a new requirements artifact, or a change to an existing requirements artifact. The process is automated for data collection and processing and hence occurs continuously, while some manual intervention, particularly in the later stages, may be required. Mapping candidate requirements to new or existing requirements arti-

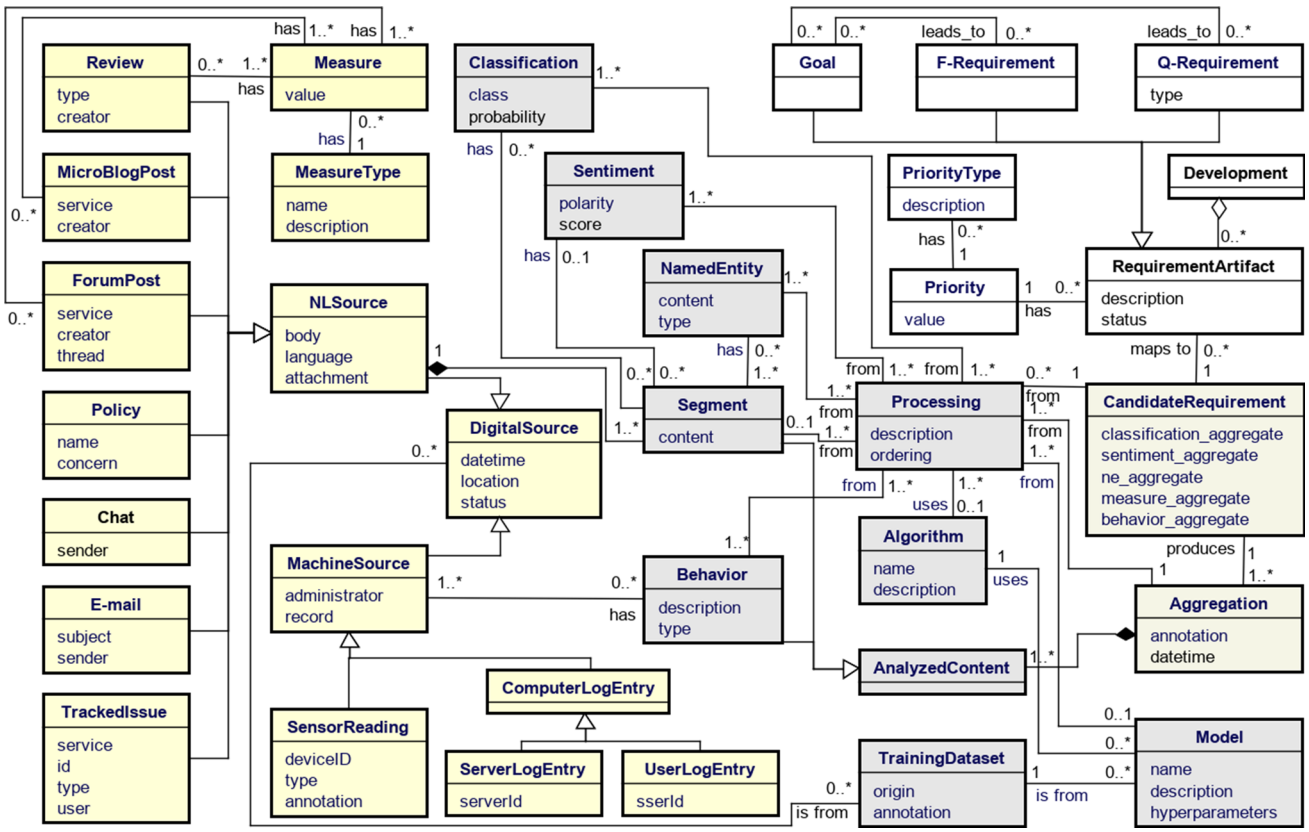

Fig. 1 Metamodel for processing and aggregating data from digital sources, and mapping them to new or existing requirements artifacts. The yellow color depicts different sources, grey for data processing, light olive for aggregation and mapping, and white for requirements

Table 1 The elements of the metamodel from Fig. 2

Concept	Description
Data sources	
DigitalSource	Digital data generated from a source created by a human or machine at a given time (*datetime*) and *location* (if known, e.g., a URL or URI); it can have a *status*, e.g., fetched or processed
NLSource	A specialization of *DigitalSource* where data is created by humans; the main data content is held in *body* written in an identified *language*, and may have some *attachment* (e.g., document, image)
MachineSource	A specialization of *DigitalSource* where data is created by a machine and supervised by an *administrator*; the main data content is held in *a record*
TrackedIssue	A specialization of *NLSource* containing in addition an *id*, a *user*, belonging to a *service*, and being part of a *thread*
E-mail	A specialization of *NLSource* with a *sender (*e-mail address) and a *subject*
Chat	A specialization of *NLSource* describing one or more text messages of communication between a *sender* and a system administrator
Policy	A specialization of *NLSource* with a policy *name* and possibly a *concern,* describing to whom it is pertaining
ForumPost	A specialization of *NLSource* with a *creator*, belonging to a forum *service*, and being possibly part of a *thread*
MicroblogPost	A specialization of *NLSource* containing in addition a *creator* and belonging to a *service* (e.g., Twitter, Tumblr, Weibo)
Review	A specialization of *NLSource* containing in addition a *creator,* designating a known or unknown (i.e., anonymous) person or role who created it and a *type*, e.g., App Review, Expert Review, Suggestion Box
Measure, MeasureType	Additional information about certain types of *NLSource,* e.g., *MicroblogPost* and *Review*, which by *value* measures the attention a digital source obtained, using a *MeasureType,* e.g., rating, re-posting, liking
SensorReading	A specialization of *MachineSource*: output of a device that detects and responds to some input (such as movement) from a physical environment
ComputerLog entry	A specialization of *MachineSource*: a record as an entry describing an occurred event within a server or software
UserLogEntry	A specialization of *ComputerLogEntry*: a record of an entry describing an event of a user having a *userId* in the user's interaction with the software
ServerLogEntry	A specialization of *ComputerLogEntry*: a record of an entry describing an event within a software at a specific server designated with a *serverId*
Data analysis	
Segment	A part of an *NLSource body*, e.g., a sentence or a paragraph (i.e., *content*). The chosen granularity depends on how the data should be analyzed in terms of NER, classification and sentiment analysis
NamedEntity	A phrase that identifies an item of interest in a *Segment*; the identified *content* belongs to a *type* (entity class), e.g., person, place, organization
Sentiment	Classification of a type of emotion or attitude expressed in a *Segment*, with a *polarity* (positive, negative, or neutral) and *score* indicating the strength of the sentiment
Classification	A description of how data from a *Segment* is grouped into a *class* (e.g., "feature request" or "bug report") and the model's associated prediction *probability*
Behavior	An analytical outcome of data generated by a machine, describing the behavior of a user or system, e.g., a frequent navigational pattern. It has a *description* and a *type*
Processing	An automated task contributing to a data transformation, which eventually produces an outcome such as *Segment, NamedEntity, Sentiment, Classification, Behavior, Model, Aggregation*, and *CandidateRequirement.* The details of a processing task are stored in *description* and several tasks may be used in an *ordering* to produce a final outcome
Algorithm	A defined set of instructions for processing data, e.g., an ML algorithm or a more basic algorithm, having a *name* and a *description*
Model	A predictive data model pertaining to a semantic scope derived from a *TrainingDataset* and a (learning) *Algorithm.* It has a *name*, *description* and *hyperparameters* used when creating the model
Training dataset	A set of data, which together with *Algorithm* (and model hyperparameters) can be used for obtaining a *Model*; if the *origin* is internal, the data comes from *DigitalSource*, while source link is stored for external data; *annotation* is the result of human labeling to support learning for some analytical task
Analyzed content	A generalization of *Segment* and *Behavior* that, after analysis, becomes part of a possible *Aggregation*

Table 1 continued

Concept	Description
Aggregation	
Aggregation	A collection of *AnalyzedContent* that is, according to *Processing*, deemed similar, e.g., refer to the same system property, and designated by *annotation*, and has a *datetime*
Candidate requirement	Analyzed and summarized information concerning a requirement, contained in *behavior aggregate*, *ne (named entity) aggregate*, *classification aggregate*, *sentiment aggregate* and *measure aggregate*. Several *Aggregation(s)* may contribute to a single CandidateRequirement over time. Mapping from *Aggregation* is determined by *Processing*
Requirements artifact	
Requirements artifact	The content obtained by mapping from a *CandidateRequirement* using *Processing* and human intervention describing a goal for the system, or some its function or a quality
Goal	A specialization of *RequirementsArtifact*, describing a desired system condition
F-requirement	A specialization of *RequirementsArtifact* describing a single system behavior (function)
NF-requirement	A specialization of *RequirementsArtifact* describing a quality of a system behavior
Priority, PriorityType	An *F-Requirement* or *NF-Requirement* is prioritized in order to determine which requirements should be developed first
Development	A list of requirements planned for development (such as requirements specification, or backlog, or even GitHub repository)

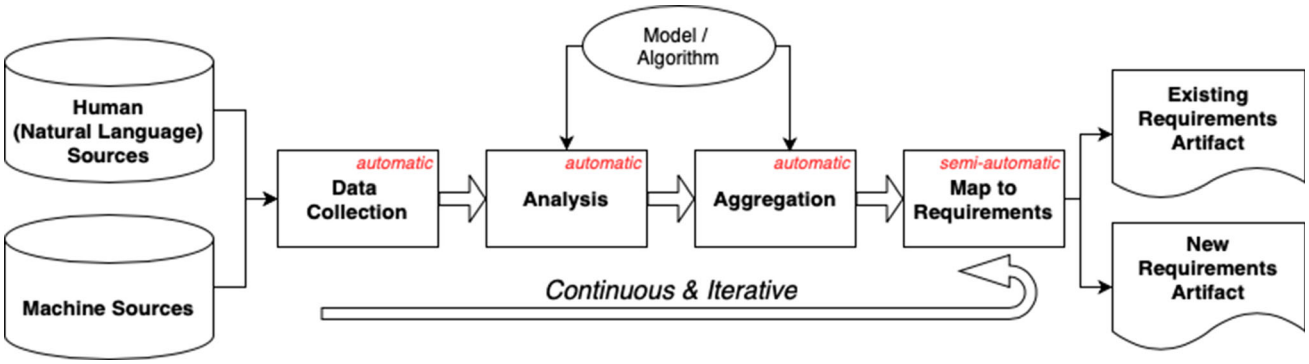

Fig. 2 The process of managing the data in an instantiation of the metamodel. The intended degree of automation of each activity is indicated in red

facts is likely to be semi-automatic, while we envision that the previous activities can be highly if not fully automated. The process is iterative in the sense that certain activities, in particular mapping to requirements, require manual engagement and decision-making that are organized by the requirements team in time-planned iterations according to the practice for change requests of the development method in place.

Data Collection The first task in the process is to identify data sources that are deemed relevant and that should be considered for requirements elicitation. The proposed metamodel (Fig. 2) guides the classification of the sources among their different types: for the sources generated by humans (*NLSource*), the data is primarily in the form of some natural language, including *E-mail*, *Review*, *MicroblogPost*, *ForumPost*, *Chat*, *TrackedIssue* from an issue-tracking system, and *Policy* from a business document. Machine-generated sources can be sensor data (*SensorReading*) or computer logs of various kinds: *ServerLogEntry* or *UserLogEntry*. Each of these source types contain some metadata, described in Table 1, which is used to programmatically fetch the data from the designated location (Fig. 2) and store the content structured according to source type. Some of the source types have *Measure* that is of *MeasureType*, which can provide insights about the relevance of data for requirements and even for prioritization in development; some examples are scores of reviews and forum posts, likes and sharing on microblogs. As a summary, the main tasks of the data collection activity are emphasized in Table 2.

Analysis The aim of this task is to provide tool-assisted data processing from a raw to a structured form such that relevant information or behavior can be identified. Fully automated analysis is critical because, for most of the sources, the data is generated at such high volumes and velocity that it is not feasible to analyze it manually, resulting in potentially valuable data being missed. Based on the classification of data source types as shown in Fig. 2, different analytical tasks to process and analyze collected data are needed.

Table 2 Tasks of data collection

Task	Guideline
Assessment of data sources to be used	Consider relevance for system's requirements Consider frequency of data emergence and amount Consider effort needed to programmatically access, fetch, and store data Consider whether persistent storage of raw data is needed
Collection from NL sources	Identify and maintain the API and the protocol of access for the source For event-driven sources, e.g., forums and microblogs, data may be collected continuously, either in near real-time or in batches at defined time intervals, from a *location*, a *timestamp* is set, and the *status* that it is fetched For task-driven sources, e.g., *Policy*, the data may be collected by a trigger based on the policy type; For sources that have *Measure*, identify its type; The fetched data is added to its processing module/program
Collection from machine sources	Identify and maintain the API and the protocol of access for the source Consider the frequency of data collection, i.e., in near real-time or in batched at defined time intervals

For *NLSource*, which is mostly unstructured, NLP is required to extract relevant information. The analytical tasks for *NLSource* include: Classification, Sentiment Analysis, and NER (included in the metamodel in Fig. 2). The outputs of these tasks are associated with a *Segment*, which allows for the *body* of an *NLSource* to be divided into smaller units, such as paragraphs or sentences, allowing for the analyses to be carried out on different granularity levels. The particular language used in the body of an *NLSource* must be known for the subsequent analysis of the data; often such information is not available as metadata, requiring the use of a pre-trained language detection tool. In order to keep track of the sequence of data transformations that have been applied to obtain a particular analysis output, these (*Classification*, *Sentiment*, *NamedEntity*) are associated with one or more *Processing*. Each processing step is, in turn, associated with an *Algorithm* or a *Model* that was used to achieve a particular data transformation. To trace how a particular model was obtained, *Model* is associated with both *Algorithm* and *TrainingDataset*, which is annotated in the case of supervised ML. The choice of data processing and analysis depends not only on the data type, but also on the structure and the availability of, e.g., ML models. One of the implications of data heterogeneity is that different prediction models are required for different types of data, irrespective of how the models are created—e.g., using ML or rule-based approaches—simply because there are likely differences in the distributions of the data that are difficult to capture in a single prediction model. For example, the language use in microblog posts versus policy documents is rather dissimilar in terms of vocabulary and grammar. This entails that if you, e.g., train a model on policy documents and apply it to tweets, it would likely perform poorly. Important to note is also that different prediction models are needed for different analytical tasks, e.g., one model is used for NER and another for sentiment analysis.

A *MachineSource* data is commonly processed to identify some form of *Behavior* (Fig. 2). By analyzing *SensorReading*, one may discover statistical anomalies that could provide information regarding needed changes in the system requirements. The case study presented in the next section would, for example, need to use NLP and ML for efficient processing and analysis of forum posts, while different techniques are needed for analyzing data obtained from the eye-tracking systems used for recognizing playing behavior patterns. By analyzing *ComputerLogEntry*, it is possible to identify the quality of a system's response in terms of performance, security, or user's behavior, such as attempts to perform interactions with the system in an unexpected, unsuccessful, or suboptimal manner.

The tools, techniques, and algorithms employed in this step of the process depend also on the desired result in relation to mapping to requirements, which in turn require different resources and skills. In order to achieve more complete data toward *RequirementArtifact*, more techniques and tools will likely need to be used, combined and developed; these would also differ depending on the structure of *RequirementsArtifact*—in our proposal, it is defined that the processed data could be classified as goals, functional or quality requirements. As a summary, the main tasks of the analysis activity are emphasized in Table 3.

Aggregation In this process step, similarities in data obtained from different digital sources are detected and grouped into an *Aggregation* that is identified by an *annotation* (numerical or symbolic). An *Aggregation* is made up of several *AnalyzedContent* instances and is hence created prior to mapping to a *CandidateRequirement* and after the *Segment* and/or *Behavior* instances have been analyzed. For example, an *Aggregation* may be composed of segmented data from reviews and microblog posts, as well as the behavior obtained from user logs, all concerning the same experience, such as missing a particular functionality. The more sources are used and the more they differ, the more difficult it is to find and aggregate those that concern the same requirement; however, this can be facilitated through automation and the use

Table 3 Tasks of analysis

Task	Guideline
Processing of NL-based source types	Determine the desired quality of automated processing Identify processing techniques, e.g., classification, sentiment analysis or NER, to apply to the collected data; Identify the availability of existing—or the possibility of developing new—models and tools that could be used for each source type, language and analytical task Decide whether and how to analyze *Measure* for the sources that have it test the outputs of processing techniques and tools in relation to the desired outcome; Continuously improve techniques and tools in the form of, e.g., adequate ML models
Processing of machine-based source types	Determine the desired quality of automated processing; Identify processing techniques for enabling rich behavior identification for each source type Identify the availability of existing—or the possibility of developing new—models and tools that could be used for each source type and analytical task Test the outputs of processing techniques and tools in relation to the desired outcome; Continuously improve techniques and tools in the form of, e.g., adequate ML models

of ML-based techniques. On the other hand, the existence of similar or related concerns in the source data means that aggregation becomes increasingly relevant for proper classification of similar concerns. Aggregation can be achieved automatically or semi-automatically through, for instance, clustering which is specified in *Processing*. An *Aggregation* is then analyzed and summarized as a *CandidateRequirement*, either by creating a new instance or by mapping to a previously created one. In the former case, the task is to, based on the content of an annotated *Aggregation*, create a proposal related to *NE_aggregate*, *classification_aggregate*, *sentiment_aggregate*, and *measure_aggregate* for NLP sources, and similarly for machine source data, i.e., based on *Behavior,* fulfill the *behaviour_aggregate* attribute in *CandidateRequirement*. To which extent aggregation can be automated depends mostly on the availability of the algorithms and models for aggregation. Aggregation can be done in numerous ways and, for example, be based on analysis outcomes alone, or also taking into account the original content. Aggregation could, for example, simply be based on the polarity values of the *AnalyzedContent* included in the annotated *Aggregation* being the same; the task may also be more complex when the information contained in several *AnalyzedContent instances* originating from a *NamedEntity*, *Behavior*, and a *Segment* would address a similar system property, yet with some different proposals related to its development, or change; here, it is on the requirements engineer to monitor the mapping and intervene by changing the outcome when the automated processing has not been correct, such that a single *Aggregation* should lead to several *CandidateRequirements* instead of one. As new data is collected and analyzed (in the two previous process steps), it could by *Processing* be placed into an existing annotated *Aggregation*, or when the data concerns a new system issue, a new *Aggregation* could be created. In both cases, the mapping to an existing or a new *CandidateRequirement* will be done as described earlier. Since the mappings may in some situations require manual engagement, the requirements engineer may decide on the pace of this task by triggering the mapping instead of it, for example, being performed continuously. As a summary, the main tasks of the aggregation activity are emphasized in Table 4.

Map to Requirements Once a new *CandidateRequirement* is instantiated, creation of a *RequirementsArtifact* of the type *Goal*, *FunctionalRequirement* or *QualityRequirement* can be attempted. The first task is to understand correctly and entirely requirement-related intention(s) from the content of the *CandidateRequirement*, which in some cases would also require reviewing the content of *AnalyzedContent*, i.e., *Segment* (for *NLSource*) and *Behavior (*for *MachineSource)*. The task is then to decide if the *CandidateRequirement* pertains to an existing requirements artifact: if the content of the

Table 4 Tasks of aggregation

Task	Guideline
Create an aggregation	Run in batches, e.g., according to some regular time interval; combine analyzed data (*AnalyzedContent*) into *Aggregations* based on which processing techniques were applied Annotate each aggregation, numerically or symbolically, for a unique identification
Map to candidate requirement	Automatically fill the attributes of a new *CandidateRequirement* from a new, i.e., non-mapped *Aggregation* Semi-automatically update the attributes of an existing *CandidateRequirement* from a non-mapped *Aggregation* when the attributes or content are similar to an existing *CandidateRequirement*, or if they are not, create a new *CandidateRequirement*

Table 5 Tasks of map to requirements

Task	Guideline
Map to requirements artifact	Map *CandidateRequirement* according to a correct requirements artifact type Combine the filled attributes of *CandidateRequirement* to create the content of a *RequirementArtifact* Use the *measure_aggregate* if filled to assess the importance of the requirements artifact
Move requirement to development	Finalize the requirements artifact according to the structure and way of working used in the RE approach in place, and move the requirement to *Development*

CandidateRequirement is assessed as entirely new or related to a change of some existing requirements artifact, then a goal or a functional or a quality requirement content should be created. This depends on the *classification_aggregate* applied in the analysis step using *Classification*, i.e., does the applied classification model match the possible types for *RequirementsArtifact*, or, if they differ, the intervention of the requirement engineer is needed. If the *measure_aggregate* is filled for a *CandidateRequirement*, that will provide additional relevant information regarding the consideration of the content, and even its prioritization. The entire step needs to be supported and reviewed by the requirement engineer, while some processing by the support of accurate ML models and algorithms could increase automation of the described tasks. Finally, the RE approach in use, i.e., whether plan-based or agile (Sect. 2.1), will influence how the requirement engineer/development team will finalize the content of the requirements artifact, including discussing possibly needed decomposition, e.g., from a goal (i.e., similar to epic in the agile approach) toward one or more functional and/or quality requirements, and once that is achieved, move the requirement(s) to *Development*. As a summary, the main tasks of the map to requirement activity are emphasized in Table 5.

4 Case study: gaming company

We have studied the applicability of the proposed framework in a case study, which is described below. Real data from the company is used to create a proof-of-concept of the proposed framework. Automated data collection and analysis, in the form of sentiment analysis and classification, is performed using an implementation of a prototype tool. We further illustrate the use of the metamodel for these and the rest of the steps of the process from Fig. 3 by instantiating the metamodel using data from the case study. For demonstration purposes, some of the activities, such as aggregation and mapping candidate requirements to requirements artifacts, are in this case performed manually. We further evaluate the proposed framework in the organizational context and conduct both introductory and evaluation sessions with the gaming company, represented by the CTO and several development team members.

4.1 Case description

The company in this case study is active in the business domain of game development, publishing and developing games. Their game portfolio consists of more than 100 titles and the company publishes its games globally, primarily through digital channels. The games are developed primarily for PC and console platforms, but also for mobile devices. The largest markets include the USA, UK, China, Germany, France, Russia and Scandinavia. Currently, they have a player base of over four million gamers that play at least one of their games each month; the total number of registered users, however, exceeds 12 million. Each game has between 1000 and 100,000 concurrent players.

The company today has around 500 employees and relies on the use of agile development methods. In order to develop games that are playful and attractive to its player base, the company is continuously making decisions about the direction of the games' evolution based on explicit and implicit user feedback. According to key persons within the organization, the main business concern is to please the requests of players regarding new features and to minimize negative feedback. Currently, analysis of collected data is to a large extent done manually, or semi-automatically at best, which requires substantial human resources. The process for analyzing the collected data is also rather ad hoc, whereby product owners and developers meet to review and discuss the data with the aim of creating request-for-change items corresponding to several user stories, or to improve existing features. However, the company has the ambition to further automate data analysis to efficiently and continuously obtain insights that will, ultimately, lead to new requirements.

4.2 Data description

We obtained data sets from one of the company's most valued data sources, i.e., reviews from the online gaming platform Steam.[1] In the proposed metamodel, this would be classified as a *NLSource* of type *Review*. Each review is timestamped and associated with a player and a specific game. There is also additional data associated with the player who posted the

[1] https://store.steampowered.com/about/

Table 6 Descriptive statistics of case study data

Game	# Reviews	# English reviews	Time period	# Tokens
Game 1	95,877	51,869	2013–08-13–2021–02-23	49.7
Game 2	133,193	69,380	2016–06-06–2021–02-23	41.2
Game 3	5879	4457	2012–06-27 – 2021–02-22	73.1

review, e.g., the number of games owned, number of reviews posted, time spent playing the game, and whether the game was purchased on Steam or received for free. Other users can also react to a posted review and this data is also associated with each review, i.e., the number of times the review was voted up or down and commented on. This metadata would correspond to *Measure* and *MeasureType* in the proposed metamodel.

We obtained reviews for three games that were collected over time (4–8 years) and, in this case, analyzed retrospectively. Descriptive statistics of these datasets are provided in Table 6.

The reviews for Game 1 and 2 are written in 28 different languages, while the reviews for Game 3 are written in 26 different languages. However, the dominant language is English, and in the automated analyses, we select only reviews written in English. While the length of reviews varies, on average the reviews contain around 41–73 tokens (tokenization is based on whitespace and punctuation and tokens include words, digits, punctuation marks etc.).

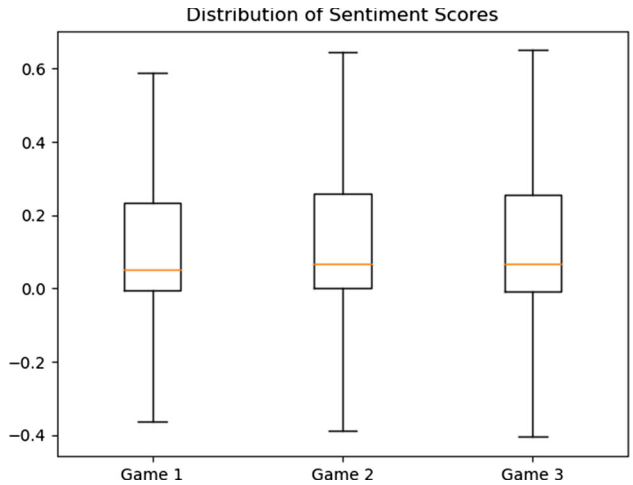

Fig. 3 The distribution of sentiment scores obtained for each of the three games

4.3 Sentiment analysis

Sentiment analysis is an NLP task to automatically and systematically identify, extract and quantify expressions of sentiment in free-text. While this may seem like a fairly straightforward task and a matter of polarity detection, in the form of positive/negative sentiment classification, it can also be approached on a deeper level that, in itself, requires several more fundamental NLP tasks to be completed. These NLP tasks can be viewed as belonging to syntactic, semantic and pragmatic layers of text analysis [30].

4.3.1 Method

Python is used for implementing the prototype and processing the data according to the metamodel (Fig. 2). The data is read into Pandas dataframes, dropping duplicates and only selecting English reviews. spaCy[2]—a Python library for NLP and large-scale information extraction tasks—in conjunction with a pre-trained NLP pipeline for English, 'en_core_web_sm', is used for processing the data. The pipeline allows for tokenization, lemmatization, syntactic parsing, tagging and NER. Sentiment analysis is conducted on the reviews for the three games using the pre-trained NLP pipeline in conjunction with the spaCyTextBlob library.[3] The sentiment analysis component takes an input text and produces scores for polarity (on a scale from -1, indicating a negative sentiment, to 1, indicating a positive sentiment) and subjectivity (on a scale from 0 to 1). We process the reviews for each of the three games, analyze the obtained sentiment scores, and select reviews with a positive and negative score, respectively.

4.3.2 Results

The distribution of sentiment polarity scores are shown in Fig. 3. As one would expect, reviews express a wide range of sentiment, from negative to positive, as well as neutral. The average sentiment score for 0.11 for Game 1, 0.12 for Game 2, and 0.10 for Game 3. While a single average sentiment score for a game may provide some indication of general player satisfaction and allows for quick comparison of games in this respect, it may be more valuable to study changes in sentiment over time. In addition, one could also identify sentiment with respect to a specific feature or aspect of a game.

[2] https://spacy.io

[3] https://spacytextblob.netlify.app

Table 7 Examples of reviews for each game with a positive or negative sentiment

Game	Polarity	Sentiment score	Review
Game 1	Positive	1.0	*Best map painting sim you'll find at the moment*
	Positive	1.0	*Lots of replayability, and just one of the best strategy games that I'd ever played*
	Positive	1.0	*THE BEST*
	Negative	− 1.0	*Terrible*
	Negative	− 1.0	*It has become horribly mired by DLC*
	Negative	− 1.0	*I hate this game…it destroyed hours of my life. HOURS!*
Game 2	Positive	1.0	*One of the best strategy games i've ever played*
	Positive	1.0	*This is one of the best ww2 sims out there*
	Positive	1.0	*One opf the best stratergy games of all time, must have for WW2 stratergy sim*
	Negative	− 1.0	*boring dont buy*
	Negative	− 1.0	*Worst AI ever*
	Negative	− 1.0	*Combat is boring. So is everything else. SHITTY!!!!!!!*
Game 3	Positive	1.0	*Awesome music*
	Positive	1.0	*That's a very welcome surprise! Keep them coming!*
	Positive	1.0	*Superb*
	Negative	− 1.0	*Awful DLC policy*
	Negative	− 1.0	*Insanely overpriced for what it gives*
	Negative	− 1.0	*Holy…..Why do I want pop music for medeival game!!!!!*

Table 7 provides examples of reviews for each game that expressed a positive sentiment, i.e., obtained a high polarity score, and reviews that expressed a negative sentiment, i.e., obtained a low (negative) polarity score. As can be seen, the games receive both very positive and very negative reviews. For game developers, it may be valuable to know why players like a specific game, which this type of automated analysis can contribute to. However, it is arguably of higher importance to analyze and act on negative reviews, in particular reviews that provide reasons for dissatisfaction.

There are also cases where a sentiment analysis model will incorrectly score or classify a review: an example of that is the following review for Game 1: "I hate this game…it destroyed hours of my life. HOURS!" This review obtained a negative sentiment score, while the review can be interpreted in a positive light, describing the game as highly addictive. Some of the other challenges involved in analyzing this type of data are also revealed by the examples, e.g., spelling mistakes, as well as abbreviated and creative language use.

4.4 Classification

Classification is the task of automatically predicting one or more classes that a set of data belongs to. Classification can be performed using rule-based systems or ML models. Classification with machine learning is typically carried out in a supervised setting, i.e., with access to (manually) labeled data.

4.4.1 Method

We use a publicly available dataset that can be used for training a classification model.[4] The data consists of app reviews from Apple AppStore and Google Play and has been manually annotated into four classes: Bug Report, Feature Request, User Experience, and Rating. The data was represented using sets of different features, and classification models were trained using a few different learning algorithms. They found that using binary classifiers for each target class—as opposed to a single multiclass classifier—led to the best results [31].

Python is used for preprocessing the training data, building classification models, and applying them to the case study data. The training data, stored in JSON format, is read into Pandas DataFrame data structure[5]; only the reviews and the manual labels are used. The Python library scikit-learn is used for feature engineering,[6] ML modeling and evaluation. Two feature representations are experimented with: bag-of-words and ngrams. A bag-of-words representation in this context entails that each review is represented as a frequency distribution of words in some vocabulary, e.g., all (lowercased) words that appear in the training data. The n-grams representation here is also a frequency distribution, but includes counts not only for unigrams but also for contiguous sequences of words

[4] https://mast.informatik.uni-hamburg.de/app-review-analysis/

[5] https://pandas.pydata.org

[6] https://scikit-learn.org

of length n; in this case $n = 2$, which means we include both unigrams and bigrams.

For each training set (corresponding to one of the target class labels) and feature representation (unigrams, unigrams + bigrams), we train a binary classifier using the random forest learning algorithm. We chose this ML algorithm for its reputation of achieving high predictive performance and its ability to handle high-dimensional data; it also has the advantage of not being overly sensitive to the choice of hyperparameters. The algorithm constructs an ensemble of decision trees, which together vote for which class label to assign to a review. Each tree in the forest is built from a bootstrap replicate of the original training set, and a subset of all features is sampled at each node when constructing the tree. This is done in order to increase diversity among the trees: as the number of trees in the forest increases, the probability that a majority of the trees makes an error decreases, as long as the individual trees perform better than random and errors are made independently. While this can only be guaranteed in theory, in practice the algorithm often achieves state-of-the-art predictive performance. In this study, we use random forest with 100 trees, while n features are inspected at each node. We evaluate the models using tenfold cross-validation and, since the classes are balanced, we employ accuracy as our primary performance metric.

Once we have evaluated the performance of the models and made a decision with respect to the choice of feature set, we re-train models using the entire training data. The models are then applied on the case study data in order to predict whether the reviews belong to one or more of the target classes: Bug Report, Feature Request, User Experience, and Rating. Note that this requires that the case study data is represented in the same fashion as was done when training the models and that the model's vocabulary is determined by the training data alone. The size of the feature set for the bag-of-words unigram models are: 2950 for Bug Report, 2650 for Feature Request, 2873 for User Experience, and 2697 for Rating. The size of the feature set for the ngram models are: 15,599 for Bug Report, 12,742 for Feature Request, 14,129 for User Experience, and 13,739 for Rating.

4.4.2 Results

The cross-validated performance with the two data representations is presented in Table 8. As can be seen, the bag-of-words representation tends to perform better than the ngrams representation, with the only exception for Feature Request where the performance of the two models is close to identical. For the sake of consistency, we employ the simpler and more efficient bag-of-words representation for all models, retrained using the entire datasets.

The overall results of applying classification models are shown in Table 9. As can be seen, a fairly large number and proportion of reviews are classified as belonging to each of the target classes. Around 34% of the reviews for Game 1 are predicted to contain bug reports, while the corresponding proportions are approximately 27% and 46% for Game 2 and Game 3, respectively. The numbers for feature requests are similar, with more proportionally more feature requests predicted for Game 3 (47%) compared to the other games (35% and 28%, respectively). The opposite is observed for ratings, with more ratings predicted for Game 1 and 2 compared to Game 3. Generally, however, a high proportion of reviews are predicted to contain ratings, which is not all that surprising. Finally, regarding user experience, around 40% of reviews are predicted to contain descriptions of the user experience, while the corresponding proportions for Game 2 and Game 3 are 33% and 53%, respectively. Note that since the case study data has not been manually labeled, we cannot make conclusions with respect to the predictive performance of the classification models on the target data: while both training and target data are in the form of reviews, they are from different sources (training data from Apple AppStore and Google Play, and target data from Steam).

Table 8 Cross-validated performance (accuracy) with two data representations

Class	Bag-of-words: unigrams	Ngrams: Unigrams + Bigrams
Bug report	74.9	74.3
Feature request	75.2	75.3
User experience	76.6	74.6
Rating	76.9	76.4

On average, 2.3 class labels were predicted for each review. To understand if there were correlations in terms of which classes were predicted for a review, we calculated the proportion of reviews across all three games that were predicted to belong to a given pair of classes. As can be seen in Fig. 4, there is a fairly large overlap between bug reports and feature requests (64.8%), as well as between bug reports and ratings (62.0%). On the other hand, the overlap is fairly small between User Experience and the other three classes (11.1% overlap with Bug Report, 10.2% overlap with Feature Request, and 9.7% with Rating).

As the classification model gives a probability that a review belongs to a given class, one can rank the reviews according to their probability of belonging to a certain class. In Table 10 are examples of reviews for which the corresponding classification model assigned a high probability to belong to the positive class.

To recap, in this section, the collected data, in the form of Steam reviews, was automatically processed and analyzed using ML for sentiment analysis and classification. The outcomes of the automated analyses contribute to the

Table 9 Classification results

Game	Bug report		Feature request		User experience		Rating	
	#	%	#	%	#	%	#	%
Game 1	17,456	33.7	17,893	35.0	20,699	40.0	32,021	62.0
Game 2	18,821	27.2	19,455	28.0	23,181	33.0	47,102	68.0
Game 3	2057	46.2	2096	47.0	2380	53.0	2170	49.0

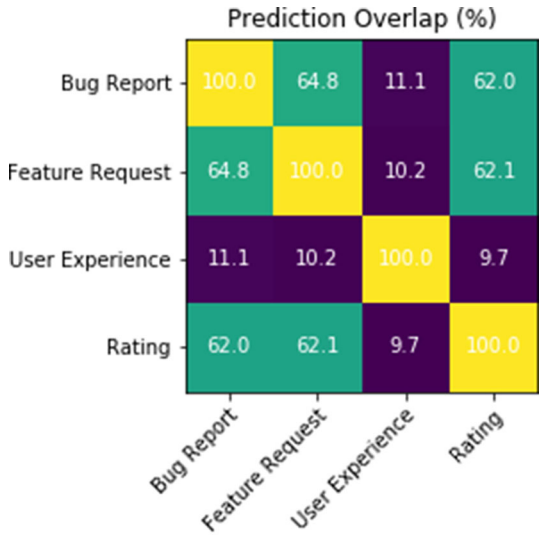

Fig. 4 Percentage of overlap between pairs of predicted classes

instantiation of the relevant parts of the metamodel, which will be demonstrated in next section.

4.5 Demonstration by metamodel instantiation

In this section, we demonstrate the use of the framework by creating instantiations of the metamodel using the case study data. In addition to the case study, other sources have also been searched in order to illustrate the ability of the framework to handle and integrate heterogeneous data sources. In the first instantiation, as shown in Fig. 5, the data analysis is fully automated using the sentiment analysis and classification models described in the previous sections. Note, however, that in this study aggregation has not been automated, but is nevertheless included to demonstrate how data is intended to be automatically aggregated to produce a candidate requirement. In the second instantiation, as shown in Fig. 6, a possible alternative scenario is demonstrated, where different choices with respect to data analysis have been made and where we show how this may have implications on aggregation and the resulting candidate requirements. Note that while the second demonstration is also based on real data from the case study, it includes certain manual processing activities (in particular, NER) for illustrative purposes.

As can be seen in Fig. 5, there are three instances of *DigitalSource*, all of type *NLSource*: one *Review* instance (r1) and two *ForumPost* instances (f1, f2). The two forum posts originate from two different forums: Reddit and a forum hosted by the case study company where players can discuss games and suggest improvements. Both forums allow posts to be liked by other forum members; in the metamodel, this is captured through *Measure* and *MeasureType*. The company's forum moreover records a reaction score for each user, which could be relevant for requirement engineering as it indicates the influence of a poster and the exposure a post is likely to get. The review is from the Steam data described earlier; the reviews can be up-voted, which indicates agreement by other players.

Each *NLSource* contains one or more segments. This allows for analyzing the free-text content as a whole, or to further split it into, for instances, paragraphs or sentences. In this case, each *NLSource* is associated with one *Segment*, which entails that the input to the subsequent processing and analysis will be the entire content. The segments are automatically analyzed through classification and sentiment analysis. While all four classification models described earlier are applied to the segments, only positive predictions are in this case retained: all three segments are classified as Feature Request, albeit with different model probabilities. Each *Classification* instance is associated with a *Processing* instance, which in this case involves the use of a machine learning model, yielding a *Model* instance. The model that predicts feature requests—as well as the classification models that predict bug reports, user experiences and ratings—is created using an *Algorithm* applied to a *TrainingDataSet*. In this case, the chosen learning algorithm is the random forest classifier, applied to an external training set, with a specific set of hyperparameters, which are attributes of the resulting model. Each *Segment* instance is also associated with a *Sentiment*, in the form of a polarity (positive or negative) and a score indicating the strength (and polarity) of the sentiment. The sentiment analysis is based on *Processing* using a pretrained *Model* for sentiment analysis. The outcome of this analysis is that one *Segment* (s1) is deemed to express a positive sentiment, while the other two (s2, s3) express a slightly negative sentiment.

In this case, all three segments are (manually) aggregated based on semantic similarity (see "random events"), which

Table 10 Examples of classification of reviews

Game	Review	Predicted class	Probability
Game 1	*Hiliariously fun yet always slightly broken. Crashes alot and patches are a double edged sword that fix some things while breaking others*	Bug report	87%
Game 2	*Everytime they decide to do a mojor DLC update it's cool, but it destroys all my saved data and takes a bit to fix until they fix the crashes XC*	Bug report	91%
Game 3	*I have tried 16 times now, and the game always crashes to desktop after a few minutes of play* *The game worked fine before (I have several hundreds of hours of game time), but this new DLC apparently broke my game* *So until that is fixed I can not even tell what's new or if it's any good… The only thing I had the time to notice, that was new, was the ability for tribes to build temples in areas with only a tribal settlement* *I have tried to start the game without the new DLC, but it doesn't work: the game still crashes to desktop after a few minutes of play* *UPDATE: The game has been updated from 2.4.1 to 2.4.2 and the problem with crash to desktop remains. It has not fixed anything in this context*	Bug report	82%
Game 1	*NOT worth it, main theme is kinda all right although I guess I expected more. Deal breaker is that you listen to the new tune then start the game and go back to original tuns! There is no on/off switch or playlist or anything—jumping from new tunes back into classical soundtrack is a no go and is very annoying in my oppinion…* *This would be worth it IF there was an option to only use new music but otherwie I wouldn't bother*	Feature request	87%
Game 2	*I absolutely love this game. If you are willing to take the time and learn how to play it is extremely rewarding. Plus all the mods make it have almost unlimited play value. My only notable complaint is the fact that the DLC is just unjustifiable. The add-ons from the DLC should have been in to begin with* *I would rank it a solid 9/10 great buy*	Feature request	86%
Game 3	*This expansion is rather meh. I get why they introduced the council to offset min-maxing with courtiers and frankly enjoy the challenge, but there should be a slight reward to bend the knee to your powerful vassals aside from them not being terribly cranky. The childhood focus thing is nice, but educating mainly results in avoiding bitter rivalries for me so it's also not very rewarding. If you apply a court teacher, some boredom sets in and you have to check a lot manually*	Feature request	88%
Game 1	*Simply not enjoyable*	User experience	100%
Game 2	*Scrumptious game:)*	User experience	100%
Game 3	*Awful DLC policy*	User experience	99%
Game 1	*This game is an absolute classic of a game. It has a steep learning curve, however, the tutorials and help tips are great for helping new players. You can choose from an extensive range of countries, or even create your own (I once created the knights templar as a country and took back their lands). This is the sort of game where you can spend hours and not even realise how long you have been playing for. I have played more hours of this game than are actually displayed*	Rating	87%
Game 2	*Very fun game. Enjoyed raping the Soviets. But game mechanics is a bit too complicated, the game keeps on crashing and multiplayer keeps on getting connection timed out. Please work on improving the game interface and multiplayer please*	Rating	88%
Game 3	*I just have played a few hours after the release, but I feel it's very fascinating* *You can run your own horse horde, making some sheeps or camp forts which can be packed to the new capital* *Plus you can make tributaries for more gold. Silk lord trade port and new fortress are also very nice new mechanic IMO* *10/10 for the lords of steppe!*	Rating	81%

could be based on both the original contents, as well as the outcome of the automated data analysis, e.g., the fact that they are all classified as Feature Request. Automatic aggregation could be achieved by unsupervised machine learning (e.g., clustering), or in a (semi-)supervised fashion if one has access to labeled data regarding which data should be aggregated. *Aggregation* could also be based on temporal alignment techniques, in particular to aggregate *MachineSource* and *NLSource* data. The *Aggregation* instance produces a *CandidateRequirement* instance, which records relevant information about the aggregation, including the outcomes of the data analysis. Here, we have aggregated information with respect to classification, sentiment analysis and measures. The measure aggregate can, for example, be leveraged for prioritization of discovered requirements. As described earlier, a candidate requirement may be mapped to a repository of existing requirements; however, in this case, we do not have access to such a repository. This phase is

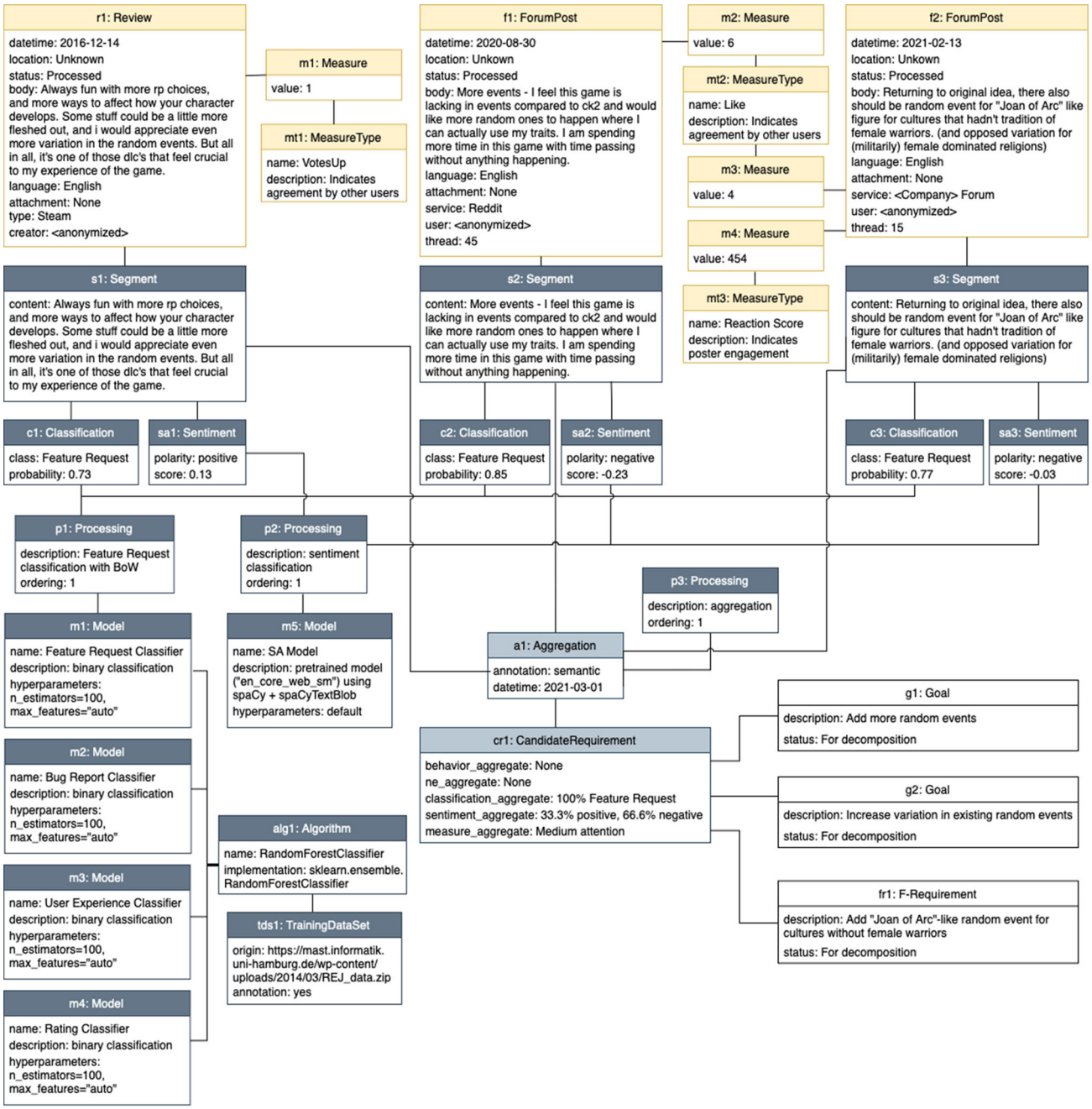

Fig. 5 Metamodel instantiation with automated processing of case study data up until aggregation

likely to involve requirements engineers or similar roles in order to conduct a deeper analysis of the automatically collected and analyzed data, before deciding on whether the candidate requirement should lead to the change of an existing *RequirementsArtifact*, be used as input for creating a new *RequirementsArtifact*, or discarded. In this case, the obtained *CandidateRequirement* may lead to the identification of two new *Goal* instances: g1 ("Add more random events" and g2 ("Increase variation in existing random events"). In addition, a functional requirement (*F-Requirement*) is obtained: fr1 ("Add 'Joan of Arc'-like random event for cultures without female warriors").

In Fig. 6, a second demonstration is shown that also includes manual NER for illustrative purposes and due to the unavailability of a publicly available model or training data for RE-specific NER. The purpose is also to illustrate how different choices may lead to different candidate requirements and, in turn, to potentially to the identification of different requirements artifacts. In this case, segmentation is carried out on the sentence level, yielding three Segment instances

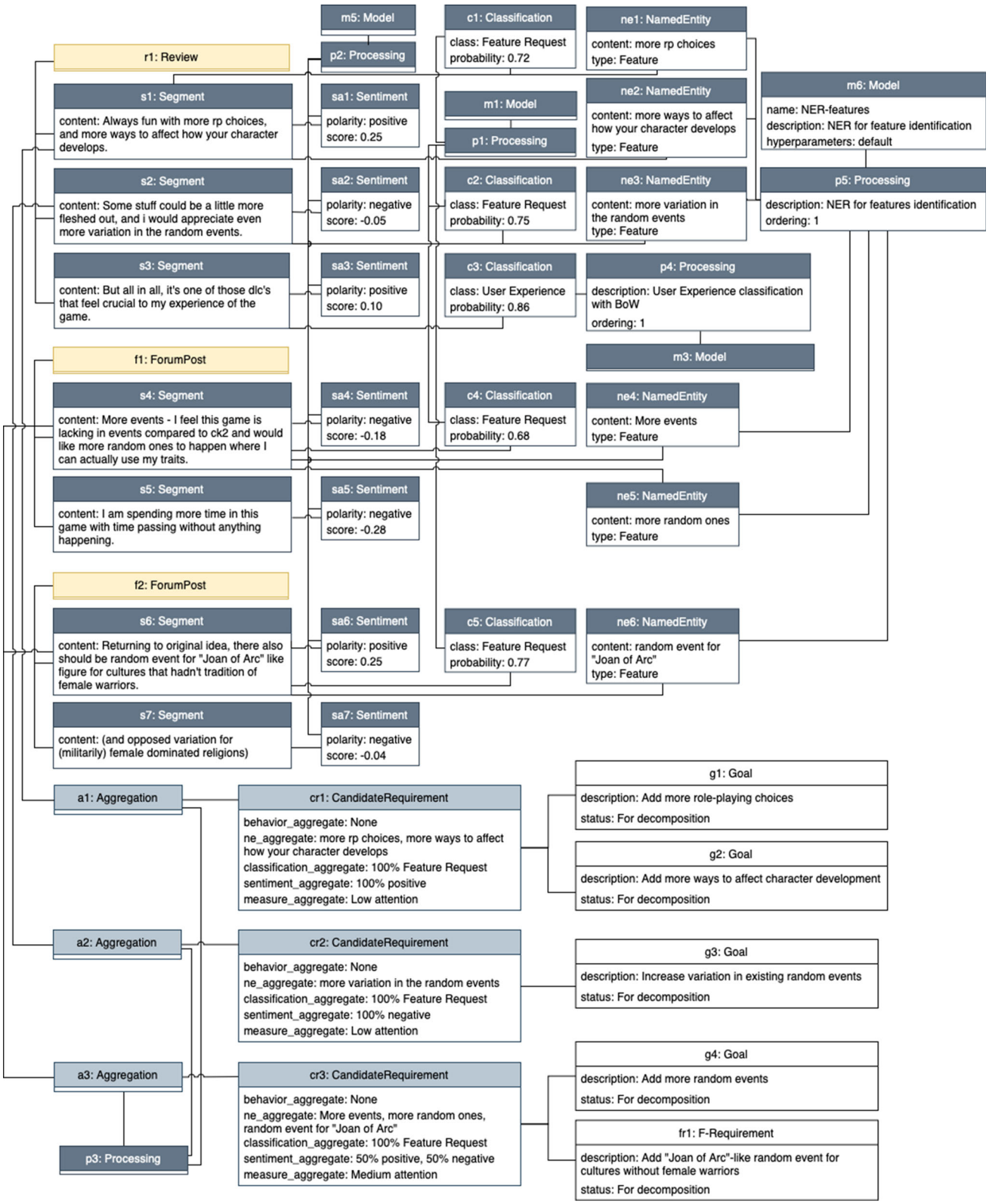

Fig. 6 Metamodel instantiation with automated and manual processing of case study data

for r1 and two *Segment* instances for f1 and f2, respectively. This entails that each segment is analyzed separately by the classification and sentiment analysis models. This may lead to the identification of more fine-grained information: for

instance, a negative sentiment for one of three segments from r1 as opposed to an overall positive sentiment for r1 in the first demonstration, and the classification of a segment as User Experience. We have also added (manual) NER in this second demonstration; in this case, the NER model tries to identify mentions of features in the segments. In s1, for instance, the following two named entities are identified: "more rp choices" and "more ways to affect how your character develops".

The choice of segmentation and analysis may have implications for aggregation. One possibility, depending of course on the aggregation model/algorithm, is that three *Aggregation* instances are created instead of just one. Here, we have one *Aggregation* instance based on s1 alone, leading to the creation of a *CandidateRequirement* instance (cr1), while another *Aggregation* instance is based on s2, leading to the creation of another *CandidateRequirement* instance (cr2), and, finally, a third *Aggregation* instance is based on s4 and s6, leading to the creation of another *CandidateRequirement* instance (cr3). The first *CandidateRequirement* instance, cr1, leads to the identification of two *Goal* instances: g1 ("Add more role-playing choices") and g2 ("Add more ways to affect character development"). The second *CandidateRequirement* instance, cr2, leads to the identification of one Goal instance: g3 ("Increase variation in existing random events"). Finally, the third *CandidateRequirement* instance, cr3, leads to the identification of one goal and a functional requirement: g4 ("Add more random events") and fr1 ("Add 'Joan of Arc'-like random events for cultures without female warriors").

4.6 Evaluation of framework with case study company

We conducted both ex ante and ex post evaluation sessions with the experts of the case study company, through interviews and demonstration. The company's representatives were represented by three employees: the CTO and two development team members. In the ex-ante period, i.e., before developing the prototype for processing the case study data sets, we carried out a series of sessions in order to demonstrate the envisioned metamodel and the method of its use; in the ex-post period, i.e., once the prototype had been developed and once the ML models had been trained and evaluated, a demonstration of the results was performed, which was followed by a discussion that we summarize below.

The company has so far invested mainly in automation of analytics related to sales, i.e., to follow the statistics of purchases across games and for different releases of games. However, they have expressed a great interest in improving their ability to monitor and act on the evolution of the requirements of its player base, particularly in order to improve performance, structuring, automation and objectivity.

In the introduction sessions, the focus was on presenting the intention and motivation of our research in order to obtain initial feedback. The conclusion of these sessions was that the company is very interested in experimental results toward automation that contribute to filtering the massive amounts of data that is generated by its player base, as currently this is only partially exploited and processed manually or semi-automatically at best. The company was particularly interested in seeing reliable evaluations concerning the performance of techniques such as sentiment analysis and classification on their data, with the ambition to apply such techniques in the near future.

In the evaluation session, we prepared several questions in a semi-structured interview. The questions provided the basis for an open discussion that followed. A summary of the interview is presented below.

Question 1 related to the overall quality and usefulness of the metamodel (Fig. 2), specifically in relation to data sources, types of data analysis, as well as aspects related to mapping of data to requirements. The feedback of the company can be summarized as follows:

- The metamodel is a very useful tool for sharing understanding of the common concepts that need to be grasped by different roles in the company (decision makers, creative directors, product owners, developers) and for different activities: for instance for the development teams of different games to unify the design architecture and programming entities and attributes; for decision makers and creative directors to obtain the same type of analytics across different games; for product owners to get more clear insights regarding available models and existing techniques for processing NL and machine-generated data.
- The types of digital data sources are well covered. Their classification is very useful as it supports separation of concerns; it also provides insights concerning the quantity and importance of data from various sources, in turn helping to better determine which data sources to focus more or less on and supporting more effective allocation of resources.
- Processing of NL sources is the most urgent and would be prioritized due to their dominant quantity. However, the company also utilizes an external machine source: an eye-tracker service for obtaining various statistics on players' behavior, i.e., concerning eye gazes, moves and fixation. Regarding other sources, such as internal and external policy documents concerning ethics and privacy, these are often consulted in relation to development of game features.
- The processing-related entities in the metamodel provide a good basis for "documenting" different techniques and predictive models.

- The aggregation and mapping to requirements elements are sufficiently elaborated. It is very useful to support identification of requirements for the Requirements Engineer or similar role with automation of certain tasks, while not suppressing the creativity of humans who created and have visions for the games.
- A possible direction in which the metamodel could be extended is to cover also sales-related metrics related to different (releases of) games in order to help with strategic prioritization of requirements, i.e., in addition to users' data that provide insights into issues and desired features.

Question 2 related to the process of using the metamodel as presented in Fig. 1 and Tables 2, 3, 4, 5. The company answered that the process is correct in terms of covered activities and emphasized that even the agile approach that has been adopted in the company in recent years needs to distinguish different tasks and steps; however, these need to be performed in a pace that facilitates a quick feedback on new releases. Since the presented process involves automation and semi-automation of various activities, it also supports development of requirements in more flexible and more rapid timeframes. This is of high value for the company because they need to have the ability to react promptly to critical issues, as well as to have the flexibility of dealing with request for changes when it is strategically the right time.

Question 3 related to the usefulness of automated sentiment analysis. The company representatives were presented with the Steam reviews datasets: amount, content, and the results of the analysis in Tables 6, 7, 8, 9, 10. The company's main feedback was the following:

- Even this narrow dataset in terms of timeframe and source confirms that the company is dealing with enormous amounts of data.
- Steam reviews have so far been seen as the main data source, but there is high interest to increase objectivity of management of players' feedback by collecting data from even other highly relevant sources, primarily from forums and Twitter.
- Performing accurate and automated sentiment analysis (which is currently not performed by the company at all) with ML models that have been optimized for each source type and tracing their use by the conceptualization (Fig. 2) has great value for i) saving human resources, i.e., the development teams that are currently reviewing the data manually, ii) largely increasing the amount of data that can be considered due to automation, and iii) decreasing the subjectivity of creative directors who are currently the ones deciding on the management of requirements (player feedback is currently manually monitored for discovering bugs and critical errors).
- It would be highly useful to automatically get accurate overviews of sentiment expressed by players with respect to (i) a specific game, (ii) a specific release of a game, (iii) a specific aspect of a game (character, feature, etc.), and to be able to track this over time.
- Negative sentiments are critical to detect and require a fast reaction when the number or impact of the players who express it is significant. Positive feedback is also important: for instance, if many players say that "female characters are great to have," then the company will probably want to incorporate more female characters in their games. Positive feedback is also relevant for understanding which new requirements and features it should invest in to attract more players to a successful game.
- Sentiment analysis would facilitate efficient granulation of relevant information and would help in timely understanding if the company is on the right track when launching a new game or a new release, as the gaming industry is very competitive: *"we can't afford anymore taking wrong directions in new releases"*.

Question 4 related to the usefulness of automated classification of the data into the following classes: Bug Report, Feature Request, User Experience, and Rating. The company confirmed the importance of the chosen classes for a specific game, a release of a game and a specific aspect of a game, assuming these were also identified automatically and accurately. "Our games are known as creative, but you can't live on these wings too long for a game if you are full of bugs or annoying features". Over time, classes should be extended to allow for "drilling down" into more sophisticated classifications, e.g., if a game is fun to play, different user experiences, and even classification of different player types (e.g., "steppe wolfs," "social" or "action-hungry") to allow for streamlining next releases to the dominant type, or even to better support the types that are under-represented in certain games.

Question 5 related to automated aggregation of the analyzed data and mapping to candidate requirements. Model instance diagrams (Fig. 5 and 6) were shown and discussed with the company, asking if this type of information would be useful to have given that automation could be achieved up to the level of *CandidateRequirement* (and possibly an attempted mapping to existing requirements), i.e., upon filtering, aggregating and summarizing large amounts of analyzed data, and then engaging a Requirements Engineer to formulate actual requirements. The main immediate feedback was highly encouraging, namely the company explained that full automation of filtering with adequate accuracy is highly welcome; yet, when it comes to aggregation and mapping to requirements, the company believes that their creative directors should have "the last word" on the requirements to ensure that they follow the established vision for a game. It is therefore important that creative directors have a view and control

on aggregation and candidate requirements; then, by combing their creative knowledge and the data from the model, they would be able to make optimal decisions for requirements and subsequent prioritization. The shown instances were understandable to the company, including the fact that the choice of segmentation, as well as analytical tasks, could have implications for aggregation and the candidate requirements, in turn affecting the resulting goals and requirements. The company also expressed that full automation should not be achieved at any price and is a process that must be implemented step by step: "It is not our goal to run after pure quantities, i.e., one should not be blind. More controlled and gradually implemented automation, with as much data as possible, is the best approach".

In the open discussion part, the company emphasized that a challenge concerns the capability to integrate "two worlds"—(i) users and (ii) software development—in a single process, while achieving both quantity and quality, having an objective basis for decision making, using dashboards to continuously monitor game conditions, and incorporating humans in the loop in the right place. Following that, the company recognized that the presented framework has that capability.

5 Discussion

We have addressed the need for a holistic approach to data-driven requirements elicitation, i.e., not only automating elicitation of requirements from digital data, but also integrating data from heterogeneous data sources. This effort aims to increase automation of the process, while considering which activities are likely to require human intervention. Successful organizations need to cater to the needs and desires of its customers or end users, and we argue that this will increasingly depend on an organization's ability to continuously collect, analyze and quickly act on information available in a variety of data sources.

In many cases, large amounts of data can be collected that may be relevant for and support requirements elicitation and other requirements engineering activities in an organization. Once the relevant digital data sources have been identified and data collection has commenced, it can be analyzed in a myriad ways. Some of the most relevant types of data analyses for requirements elicitation have been captured by the proposed metamodel, such as sentiment analysis, classification, and NER for natural language data, while one typically wants to identify "behavior" when analyzing data generated by machines. We do not claim the metamodel with respect to possible data analyses to be exhaustive; additional types of data analyses can readily be incorporated into the existing metamodel. In this study, we demonstrated the use of ML for sentiment analysis and classification of natural language data in the form of reviews and forum posts. Sentiment analysis is highly relevant for identifying end users' sentiments with respect to a software system in general, a specific aspect of a system or even in relation to particular feature. The power and utility of sentiment analysis does not primarily lie on an instance level, i.e., for a specific review or forum post, as this can readily and more accurately be performed by a human being. However, sentiment analysis is useful in that it can be carried out efficiently on a large scale, providing insights about the overall sentiment expressed by a user base, or groups of users, over time. An organization may, for instance, be particularly vigilant after releasing a new software update. This allows an organization, in the case study a gaming company, to readily obtain on overview of player satisfaction over time and work to improve its games based on feedback from players. A similar argument can be made for automatic classification of natural language data. In this study, we developed a ML model to classify reviews into different classes: Bug Reports, Feature Requests, User Experiences, and Ratings. While classifying individual reviews can help to identify reviews that are worthy of attention from a requirements engineer or similar role, and aid in automatic aggregation of data, it is also useful for efficiently getting an overview of an organizations' products. For the company in the case study, for instance, automatic classification can provide an overview of the number of bugs that are reported for each of its games. Continuously identifying reported bugs—even in data that is not necessarily aimed at the organization responsible for the product—allows an organization to react quickly and thereby increase customer satisfaction.

A relevant discussion concerns the chosen level of specialization for the metamodel (Fig. 2); in our previous study [5], we focused on the agile RE approach, as well as in [36]. However, we have received feedback regarding this focus on agile RE suggesting that the metamodel and the process could instead be proposed in a way that supports both agile and conventional RE methodologies; this is also supported by empirical studies according to which the two approaches are often mixed in practice, especially in large-scale agile development [11, 12]. We have therefore in this study generalized the part of the metamodel related to requirements. We conducted several unstructured interviews with international academics in the RE community, as well as development companies in Sweden. In summary, the response was in favor of making the metamodel generic and then tailored to a specific domain, development approach, etc., depending on the usage context. The requirements are specific only in the way that they are based on natural language, either following the traditional template or that of user stories in the agile methodology. In other words, they are not tied to any software model because the ambition of the study was to, from big data, reach the point of requirements explication. Once a require-

ment has been described, it is not different from requirements described by traditional stakeholders. Further development-related steps, e.g., (automated) mapping of NL requirements to use cases, some design entities, or to code, are necessary; this has been explored by several studies in the context of NLP for requirements identification.

Regarding the relevance of the framework, the main parameters pertain to the diversity of digital data sources used, and the amount of generated data. Not all companies use the same types of digital data sources and probably no company uses them all. Hence, a company needs to assess the relevance of the data sources toward requirements elicitation and whether the data is of sufficient quantity and quality to aim for automated or semi-automated processing. Besides the gaming company in the case study, which has around twelve million players, other organizations have also confirmed the need for and value of such a framework, e.g., a bank with up to one million customers. Moreover, a number of universities with 10–100 thousand students have expressed the need for classifying different sources of student feedback regarding online education platforms and techniques for processing data from these sources.

The assessment of the framework in this study has been applied on external systems (games). Regarding the applicability of the proposed framework for internal systems, in [5] we used for the illustration an internal system, i.e., a custom-developed web application for managing research proposals of Stockholm University employees; additional research could specifically target internal systems by starting from a feasibility assessment regarding which of the tasks are relevant to invest in automating, as well as determining which should be semi-automated, and accordingly making a model of relevant ML models and algorithms. In this study, we choose to study the gaming company because its data is much larger, global and attractive, while the company is already struggling with the fact that they are not able to adequately process the enormous amounts of heterogeneous data that they have access to.

The focus of this study has been on further developing the foundations for a holistic and data-driven framework for requirements elicitation from heterogeneous data sources. However, while there have been many efforts in the literature to analyze explicit and implicit user feedback, there is a need for more research and development of data analysis methods in the RE domain. If the predictive performance of the models used for automatic analysis of data is insufficient, efforts to integrate data from different sources are severely hampered. Today, data analysis increasingly relies on the use of ML, often in a supervised setting, which requires access to labeled data. Moving forward, it will be important that created resources, e.g., in the form of annotated datasets and even ML models, are made publicly available to the research community. For this study, for example, we were unable to find labeled data or ML models for NER tailored to requirements elicitation.

In future work, it will also be important to evaluate and possibly develop novel methods for aggregation of data for the purposes of creating candidate requirements, as well as for mapping between a candidate requirement and a repository of existing requirements. In the case study, aggregation and mapping of candidate requirements to requirements artifacts were performed manually with the purpose of illustrating the ambition of the proposed framework for data-driven requirements elicitation and the process for its use. We envision that aggregation is an activity that can be largely automated using, e.g., clustering and other techniques based on semantic similarity. It will be important for future research to tackle this challenge, in particular to propose techniques that are able to aggregate language data with machine-generated data. Mapping candidate requirements to requirements artifacts will most likely remain semi-automatic at best, in the sense that an automated mapping could be attempted between a candidate requirement and a repository of existing requirements artifacts; however, a requirements engineer or similar role will most likely be required to verify a proposed mapping, or to select the best match among a number of candidates. Depending on the form of the requirements artifact, it is also plausible that human involvement may be needed to formulate requirements. Another relevant issue here is how automation, e.g., in the form of classification, could be achieved in order to facilitate correct identification of the requirements artifact type.

The validation of the proposed framework in the case study was partial in the sense that not all classes in the metamodel were instantiated. While it is not necessary, or even very likely, that all classes would be instantiated when used in a particular context—it may, for example, be decided that only a subset of the available types of digital sources are relevant—it would be important to validate other parts of the proposed framework, most notably the use of machine-generated data, in future work. However, while some research efforts have been made, more research is also required for developing and evaluating techniques that can derive requirements or requirements-related information from machine-generated data. This is naturally a more complex challenge compared to exploiting natural language data since there are relatively more steps to go from machine-generated data to some requirements representation. We have nonetheless included machine sources in the proposed framework as the ambition is for it to be holistic, and we believe that such data will play an important role in the future of data-driven requirements elicitation.

6 Conclusions

In this study, we have extended the work presented in [5], wherein the foundations were laid for a novel framework, consisting of a conceptual metamodel and the method for its use, that supports data-driven requirements elicitation from heterogeneous data sources. In addition to further elaborating the metamodel and the process to support the proposed framework, we have conducted an in-depth case study using real data from a large gaming company—a company that has identified several relevant data sources for analyzing feedback and behavior from the games' player bases, but is currently carrying out related activities in a rather ad hoc manner and still relying, to a large extent, on manual analysis. The case study has allowed us to further improve and validate the metamodel and to demonstrate the feasibility of the framework, including the use of fully automated data analysis, with real data.

The motivation for the study is the need to provide business organizations with a systematic aid for dealing with the complexity of digital data management: available and prioritized data sources, amount and dynamics of the data in these sources, and different techniques for data processing and analysis. The framework is intended to complement existing approaches to requirements elicitation.

The main direction for future work concerns further elaboration of automation and efficiency of the last phases of the process, such as mapping to requirements, as well as classifying the use of processing techniques and algorithms according to the specifics of data sources and software product to which they concern, and in accordance to that classify different situational methods [32] and capabilities [33] for elicitation. Another direction of interest is a further development of the framework for model-driven engineering, i.e., to leverage abstraction and automation in software development [34, 35].

Funding Open access funding provided by Stockholm University.

References

1. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In Proceedings of the Conference on the Future of Software Engineering (ICSE), pp. 35–46. ACM Press, New York (2000)
2. Pohl, K.: Requirements engineering: fundamentals, principles, and techniques. Springer, Heidelberg, New York (2010)
3. Quer, C., Franch, X., Palomares, C., Falkner, A., Felfernig, A., Fucci, D., Maalej, W., Nerlich, J., Raatikainen, M., Schenner, G., Stettinger, M., Tiihonen, J.: Reconciling Practice and Rigor in Ontology-based Heterogeneous Information Systems Construction. In Proceedings of the Practice of Enterprise Modeling (PoEM), LNBIP vol.335, pp. 205–220, Springer (2018)
4. Malej, W., Nayebi, M., Ruhe, G.: Data-Driven Requirements Engineering – An Update. In Proceedings of Int. Conference on Software Engineering: Software Engineering in Practice (ICSE SEIP), IEEE Press (2019)
5. Henriksson A., Zdravkovic J.: A Data-Driven Framework for Automated Requirements Elicitation from Heterogeneous Digital Sources. Proceedings of The Practice of Enterprise Modeling. PoEM. Lecture Notes in Business Information Processing, vol 400. Springer, pp. 351–365 (2020)
6. Dalpiaz, F., Niu, N.: Requirements engineering in the days of artificial intelligence. IEEE Softw. **37**(4), 7–10 (2020)
7. Dąbrowski, J., Letier, E., Perini, A., Susi, A.: Mining User Opinions to Support Requirement Engineering: An Empirical Study. In Proceedings of International Conference on Advanced Information Systems Engineering, LNCS, vol. 12127. pp. 401–416, Springer, Berlin (2020)
8. Cohn, M.: User Stories Applied: for Agile Software Development. Addison Wesley, Redwood City (2004)
9. van Vliet M., Groen E.C., Dalpiaz F., Brinkkemper S.: Identifying and classifying user requirements in online feedback via crowdsourcing. In Requirements Engineering: Foundation for Software Quality, REFSQ. Lecture Notes in Computer Science, vol. 12045. pp. 143–159. Springer, Berlin. (2020)
10. Kirikova, M.: Continuous requirements engineering. In Proceedings of International Conference on Computer Systems and Technologies (CompSysTech), pp. 1–10, ACM DL (2017)
11. Kasauli, R., Knauss, E., Horkoff, J., Liebel, G., de Oliveira Neto, F.: Requirements engineering challenges and practices in large-scale agile system development. J. Syst. Softw. **172**, 110851 (2020)
12. Koutsopoulos, G., Kjellvard, N., Magnusson, J., Jelena Z.: Towards an integrated metamodel for requirements engineering. In Proceedings of FIP Working Conference on The Practice of Enterprise Modelling, Forum, ceur-ws, vol 2586, pp 40–53 (2020)
13. Meth, H., Brhel, M., Maedche, A.: The State-of-the-Art in Automated Requirements Elicitation. Information and Software Technology, vol. 55, pp. 1695–1709, Elsevier, Amsterdam (2013)
14. Manrique-Losada, B., Zapata-Jaramillo, C. M., Burgos, D. A.: Re-Expressing business processes information from corporate documents into controlled language. In Natural Language Processing and Information Systems, pp. 376–383, Springer, Berlin (2016)
15. Nogueira, F. A., De Oliveira, H. C.: Application of Heuristics in Business Process Models to Support Software Requirements Specification. In Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS), vol. 2, pp. 40–51 (2017)
16. Oriol, M., Martínez-Fernández, S., Behutiye, W., Farré, C., Kozik, R., Seppänen, P., Vollmer, A.M., Rodríguez, P., Franch, X., Aaramaa, S., Abhervé, A., Chorąś, P.J.: Data-driven and Tool-supported Elicitation of Quality Requirements in Agile Companies. Softw. Qual. J. **28**(3), 931–963 (2020)
17. Franch X, et al. Data-driven elicitation, assessment and documentation of quality requirements in agile software development. In

International Conference on Advanced Information Systems Engineering. Springer, Cham (2018)
18. Xiao, M., Yin, G., Wang, T., Yang, C., Chen, M.: Requirement Acquisition from Social Q&A Sites. In Proceedings of 2nd Asia Pacific Symposium (APRES), vol. 558, pp 64–74 (2015)
19. Dhinakaran, V. T., Pulle, R., Ajmeri, N., Murukannaiah, P. K.: App Review Analysis via Active Learning: Reducing Supervision Effort without Compromising Classification Accuracy. In Proceedings of 26th Interantional Requirements Engineering Conference (RE), pp. 170–181, IEEE (2018)
20. Williams, G., Mahmoud, A.: Mining Twitter feeds for software user requirements. In Proceedings of 25th International Requirements Engineering Conference (RE), pp. 1–10 IEEE (2017)
21. Morales-Ramirez, I., Kifetew, F. M. Perini, A.: Analysis of Online Discussions in Support of Requirements Discovery. In Proceedings of Interanational. Conference on Advanced Information Systems Engineering (CAiSE), vol. 10253 LNCS, pp. 159–174, Springer, Berlin (2017)
22. Xie, H., Yang, J., Chang, C.K., Liu, L.: A statistical analysis approach to predict user's changing requirements for software service evolution. J. Syst. Softw. **132**, 147–164 (2017)
23. Voet, H., Altenhof, M., llerich, M., Schmitt, R. H., Linke, B.: A framework for the capture and analysis of product usage data for continuous product improvement. J. Manuf. Sci. Eng. ASME 141: 021010 (2019)
24. Oriol, M., Stade, M.J.C., Fotrousi, F., Nadal, S., Varga, J., Seyff, N., Abelló, A., Franch, X., Marco, J., Schmidt, O.: FAME: Supporting continuous requirements elicitation by combining user feedback and monitoring. In Proceedings of the 26th International Requirements Engineering Conference (RE), pp. 217–227, IEEE Computer Society (2018).
25. Wüest, D., Fotrousi, F., Fricker, S.: Combining monitoring and autonomous feedback requests to elicit actionable knowledge of system use. In Proceedings of the 25th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), pp. 209–225, LNCS 11412 (2019)
26. Lim, S., Henriksson, A., and Zdravkovic, J.: Data-driven requirements elicitation: a systematic literature review. Springer Nature Computer Science, Vol 2/16 (2021)
27. Wang, Z., et al.: A novel data-driven graph-based requirement elicitation framework in the smart product-service system context. Advanced Engineering Informatics (2019)
28. Franch, X., Ralyté, J. Perini, A., Abelló, A., Ameller, D., Gorroñogoitia, J., Nadal, S., Oriol, M., Seyff, N., Siena, A., and Susi, A.: Situational Approach for the Definition and Tailoring of a Data-Driven Software Evolution Method. In Proceedings of 28th CAiSE, LNCS, vol. 10816, pp. 603–618, Springer, Berlin (2018)
29. Henriksson, A.: Learning multiple distributed prototypes of semantic categories for named entity recognition. Int. J. Data Min. Bioinform. **13**(4), 395–411 (2015)
30. Cambria, E., Poria, S., Gelbukh, A., Thelwall, M.: Sentiment analysis is a big suitcase. IEEE Intell. Syst. **32**(6), 74–80 (2017)
31. Maalej, W., Kurtanović, Z., Nabil, H., Stanik, C.: On the automatic classification of app reviews. Requir. Eng. **21**(3), 311–331 (2016)
32. Mirbel, I., Ralyté, J.: Situational method engineering: combining assembly-based and roadmap-driven approaches. Requir. Eng. Journal **11**(1), 58–78 (2006)
33. Zdravkovic, J., Stirna, J., Kuhr, J.-C., Koç, H.: Requirements engineering for capability driven development. In: Frank, U., Loucopoulos, P., Pastor, Ó., Petrounias, I. (eds.) The Practice of Enterprise Modeling, pp. 193–207. Springer, Berlin, Heidelberg (2014)
34. Pastor, O.: Model-driven development in practice: from requirements to code. In Proceedings of SOFSEM '17: Theory and Practice of Computer Science. Lecture Notes in Computer Science, vol. 10139. Springer, Cham (2017)
35. Bucchiarone, A., Cabot, J., Paige, R.F., et al.: Grand challenges in model-driven engineering: an analysis of the state of the research. Softw. Syst. Model. **19**, 5–13 (2020)
36. Franch, X., Henriksson, A., Ralyté, J., Zdravkovic, J.: Data-Driven Agile Requirements Elicitation through the Lenses of Situational Method Engineering. To appear In Proceedings of the 29th International Requirements Engineering Conference (RE): IEEE Digital Library (2021).

Aron Henriksson is an Assistant Professor at the Department of Computer and Systems Sciences (DSV) at Stockholm University. He holds a Ph.D. in Computer and Systems Sciences from Stockholm University, and his dissertation was awarded the 2016 Börje Langefors Prize for best PhD thesis in Informatics at a Swedish university. Aron's research is primarily in the area of natural language processing, focusing on semantic representations and language models. His research includes applications of natural language processing and machine learning to healthcare and, lately, data-driven requirements engineering.

Jelena Zdravkovic is a Professor at the Computer and Systems Sciences (DSV) department at Stockholm University. She has a Ph.D. in Computer and Systems Sciences at The Royal Institute of Technology (KTH), as well as the MBA degree in E-commerce. Jelena's research activities include requirements engineering and capability-driven IS development. Jelena has published around 100 refereed papers in international conferences and scientific journals on the topics of requirements engineering and capability-driven development. She is in the Editorial Board of Springer BISE and RE Journals, as well as a regular reviewer and guest editor for a number of other international journals including several of Springer, Elsevier's Journal of Systems and Software and Information and Software Technology Journal, and IEEE Computing. Jelena has organized a number of international conferences and workshops in the IS Engineering discipline, and she serves in the program committees of many of them. Her latest interest includes Data-driven Requirements Engineering, as well as development and analysis of Digital Business Ecosystems.