A Data-Driven Framework for Automated Requirements Elicitation from Heterogeneous Digital Sources

Aron Henriksson, Jelena Zdravkovic

▶ **To cite this version:**

Aron Henriksson, Jelena Zdravkovic. A Data-Driven Framework for Automated Requirements Elicitation from Heterogeneous Digital Sources. 13th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling (PoEM 2020), Nov 2020, Riga, Latvia. pp.351-365, 10.1007/978-3-030-63479-7_24 . hal-03434649

HAL Id: hal-03434649

https://inria.hal.science/hal-03434649v1

Submitted on 18 Nov 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

A Data-Driven Framework for Automated Requirements Elicitation from Heterogenous Digital Sources

Aron Henriksson and Jelena Zdravkovic

Department of Computer and Systems Sciences, Stockholm University,
Postbox 7003, 164 07, Kista, Sweden
{aronhen, jelenaz}@dsv.su.se

Abstract. Increased digitalization and the pervasiveness of Big Data, along with vastly improved data processing capabilities, have led to the consideration of digital data as additional sources of system requirements, complementing conventional stakeholder-driven approaches. The volume, velocity and variety of these digital sources present numerous challenges which existing system development methods are unable to manage in a systematic and efficient manner. We propose a holistic and data-driven framework for continuous and automated acquisition, analysis and aggregation of heterogeneous digital sources for the purposes of requirements elicitation and management. The proposed framework includes a conceptualization in the form of a meta-model and a high-level process for its use; the framework is illustrated in a real case of an enterprise software.

Keywords: Enterprise Modeling, Data-Driven Requirements Engineering, Meta-Model, Natural Language Processing, Machine Learning, User Stories

1 Introduction

Requirements elicitation is traditionally stakeholder-driven, where the primary sources of information are generated through interviews with business experts [1, 2]. Due to business and technology changes, intensive industry competition, as well as changing needs and expectations of customers, it is not possible to fully determine the desired behavior of a software system prior to development; instead, it is often preferable to build an initial solution fast, which is later evolved over time. Although agile methodologies have contributed to this by introducing development practices that facilitate interactive and timely software delivery [3], it has become relevant to enable the means for eliciting requirements from a broader spectrum of sources [4].

Increased digitalization has led to continuous generation of large amounts of digital data, both in organizations and in society at large. In the requirements engineering (RE) community, there has been a growing interest in considering digital data as additional sources for requirements acquisition [4,5]. Considering digital data for requirements acquisition is important for a number of important reasons: (i) it allows for increased scalability and to consider a wider scope of stakeholders, including potentially large and dispersed user bases (ii) it facilitates increased automation, both of elicitation and subsequent RE activities (iii) it makes continuous elicitation possible, supporting

software evolution through more frequent software improvements. There is 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 – and often *non-intended*, i.e. data not originally created for the purposes of requirements elicitation, e.g. various social media and online user reviews.

Although their potential as sources of information for system requirements is becoming widely recognized, there are numerous challenges in processing the data effectively such that it can be fully exploited in organizations for development and evolution of enterprise software [6]. In contrast to conventional methods, where requirements are elaborately elicited and guided by requirements engineers in stakeholder interviews, these digital sources are typically not explicitly created for the purposes of requirements elicitation and therefore limited in terms of completeness. Moreover, since the data tends be either (i) human-generated, in the form of unstructured natural language or (ii) machine-generated, e.g. sensor data or computer logs, and thus only providing implicit feedback on requirements, there is an inherent challenge in transforming the obtained raw data into some canonical requirement format, which is understandable to a software team and feasible to develop and implement, such as User Stories [7]. Finally, 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 not methodologically considered in agile approaches that are mainly interaction driven. The motivation for this research is thus to overcome limitations of current approaches concerning utilization of data from digital sources for system requirements. By taking a holistic view, we propose a data-driven framework and model-based approach by means of an integrated meta-model for continuous acquisition, structuring, and aggregation of digital data, for obtaining new, or updating existing, system requirements. The proposed framework is intended to bring a contribution to a continuous run-time evolution of existing enterprise software systems by setting up a conceptual basis for linking digital data referring to a given software and requirements artifacts, as well as the means of their processing.

2 Background

2.1 Conceptual Models for Requirements Engineering

RE activities traditionally start with *elicitation* of information from available sources, primarily human stakeholders. The collected information needs to be *documented* according to a canonical requirement artifact format [2], such as the recently prevailing User Story [3, 7], originating from agile system development. Moreover, documented requirements need to be *analyzed* ("negotiated") for importance, dependencies, conflicts, and prioritized accordingly. *Validation* is performed to ensure the requirement specification is consistent, complete, and meets stakeholders' needs; in User Story driven RE, this is done iteratively: documented User Stories are placed in a "product backlog", i.e. a prioritized list of User Stories and their related tasks [7].

Conceptual models are mainly used in RE for defining a modeling language and are often referred to as meta-models. Their main purpose is to represent requirements' related syntax and semantics pertaining to information systems; as such, meta-models for RE have been the subject of numerous research studies. Some of them have a general purpose, such as representation of the main elements, their meanings, and relationships among them, for traditional/plan-driven [2] or agile/User Story oriented RE [7]. Others set the focus to a particular business domain of interest, or to a concern, e.g. goals, traceability, security, embedded-systems or systems families.

With the emergence of Big Data, the meta-models have increasingly focused on enabling classification, interoperability, and automation of RE-related tasks [4,8,9]. Being focused on goals and traditional requirements, none of the approaches have mapped digital sources to entire User Story artifacts, nor have they classified the means of processing different sources, or to distinguish possible aggregations and outcomes.

2.2 Data-Driven Requirements Elicitation

There have been many efforts to automate requirements elicitation from various sources. These can be divided according to the type of sources that are used. One research track has focused on eliciting requirements from *static* data sources, typically domain knowledge [10], e.g. business documents [11] or various types of models [12, 13]. In some efforts, the goal is to convert existing requirements, elicited through conventional elicitation techniques, into some other form (e.g. from a model to textual descriptions or vice versa) [14], or to automate subsequent RE activities based on existing requirements [15]. A research track concerns eliciting requirements from more *dynamic* data sources, motivated by the increasing prevalence of user feedback. Common sources include online reviews [16], e.g. app reviews, microblogs [17], discussion forums [18] and mailing lists [19]. While this data is created by humans and in natural language, there have also been a few attempts to mine machine-generated sources, such as usage data [20] or sensor data [21]. However, most efforts have so far focused on identifying and classifying requirements-related information in a single source [22] and have aggregated information from multiple (types of) sources. Recently, more holistic views on data-driven requirements elicitation have been presented [4,5]; however, the studies have not elaborated the provided high-level views.

Since the data is mainly in the form of free text, natural language processing (NLP) is often applied in order to extract requirements information. Identification of information in free text is a task referred to as Named Entity Recognition (NER). NER concerns classification of sequences of words [23], where, here, the goal is to identify the beginning and end of any type of predefined requirements-related information, e.g. mentions of features. There are two broad approaches to classification: rule-based approaches and machine learning (ML). Rule-based approaches involve domain experts manually constructing rules to identify and classify requirements-related information. ML involves using an algorithm to learn from data to perform the classification. This learning process is typically supervised and relies on manual annotation of examples. This is costly, but once an ML model has been created, it can be applied repeatedly to facilitate automation of requirements elicitation.

3 Framework for Integration of Data to Requirements

3.1 Meta-Model for Data-Driven Requirements Elicitation

To achieve a holistic solution for integration of heterogeneous digital data sources for software requirements discovery, we have envisioned a data-driven framework including a meta-model and a methodology for its use. The main concern of this study is to design a detailed meta-model (Figure 1), and to describe a basic process for its use.

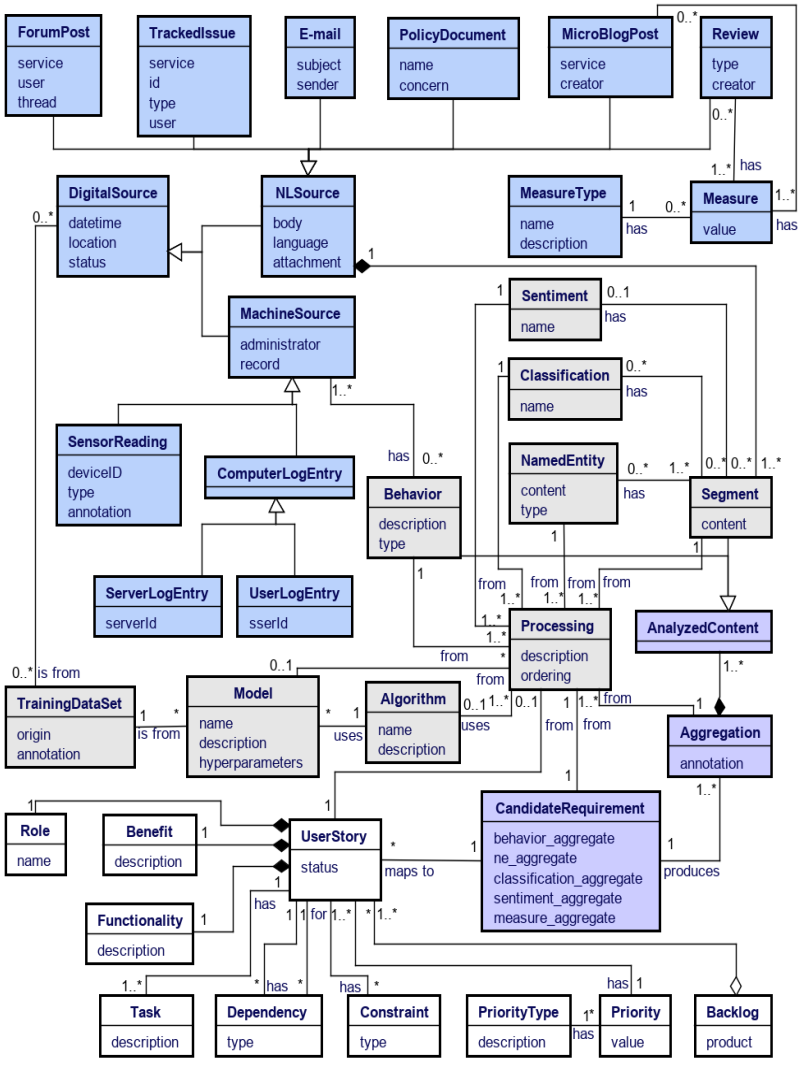

Figure 1. Meta-model for processing and aggregating data from digital sources, and mapping them to new or existing requirements artifacts.

The meta-model distinguishes: a classification of digital data sources (in blue), the elements for processing the data (in gray), the elements for data aggregation and further mapping to requirements (in purple), and a conceptualization of the User Story requirements artifact and related elements (in white). The depicted classes and relationships are in detail described in Table 1 below:

Table 1. The elements of the integrated meta-model from Figure 1.

Class	Description
DigitalSource	Digital data generated from a source created by a human or machine at a given time and location (if known); 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).
Machine Source	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*.
E-mail	A specialization of *NLSource* with a *sender (*e-mail address) and a *subject*.
ForumPost	A specialization of *NLSource* with a *creator*, belonging to a forum *service*, and being possibly part of a *thread.*
Microblog Post	A specialization of *NLSource* containing in addition a *creator* and belonging to a *service* (e.g. Twitter, Tumblr, Weibo).
Policy Document	A specialization of *NLSource* with a policy *name* and possibly a *concern,* describing to whom it is pertaining.
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.
Tracked Issue	A specialization of *NLSource* containing in addition an *id*, a *user*, belonging to a *service*, and being part of a *thread.*
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.
Sensor Reading	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.
UserLog Entry	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.
ServerLog Entry	A specialization of *ComputerLogEntry*: a record of an entry describing an event within a software at a specific server designated with a *serverId*.
Segment	A component of an *NLSource body*, e.g. a sentence or a paragraph, or the entire *body*. 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 the possible *value*: positive, negative, or neutral.
Classification	A description by *name* of how data from a *Segment* are grouped into a class, e.g. "feature request" or "bug report".
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 *User Story*. A processing task's details 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* and part of a possible *Aggregation*.
Aggregation	A collection of *AnalyzedContent* that is, according to *Processing*, deemed similar, e.g. refer to the same functionality, and designated by *annotation*.
Candidate Requirement	Analyzed and summarized information concerning a requirement, contained in *behavior aggregate*, *ne aggregate*, *classification aggregate*, *sentiment aggregate* and *measure aggregate*. Several *Aggregation(s)* may contribute to this over time. Mapping from *Aggregation* is determine by *Processing*.
UserStory	Requirement artifact obtained by mapping from a *CandidateRequirement* using *Processing(s)* and human intervention for describing a system function from a user's perspective, and consisting of three parts: *Role*, *Functionality* and *Benefit*. Role is a type of user; *Functionality* represents a desired system behavior. *Benefit* is the advantage provided by the system function to the user.
Task	A broken-down part of *UserStory* functionality contributing to the completion of the story.
Constraint	A qualitative aspect that must be considered for one or more *UserStory*, e.g. performance and reliability.
Dependency	A relation between multiple *UserStory* meaning that if execution of a User Story 2 needs User Story 1 based on a dependency *type*, then User Story 1 must first be developed.
Priority, PriorityType	Each *UserStory* is prioritized in order to determine which stories should be first developed.
Backlog	A number of *UserStory*, ordered by *Priority*, to be developed in an iteration.

3.2 Automated Requirements Elicitation Process

The basic process according to which the meta-model is instantiated is illustrated in Figure 2. The main four activities in the process are: (1) data collection, (2) data processing and analysis, (3) aggregation of analyzed data from one or more sources into candidate requirements, and (4) mapping of the candidate requirements to user stories. The inputs to data collection are various, potentially heterogeneous sources of data, generated primarily by humans and machines. The output of the process is a potential match to an existing user story, or as a candidate for the creation of a new user story. The entire process is highly iterative, where data is continuously collected and processed, either in (near) real-time or in batches according to some regular time

intervals. The process is intended to be automated to a great extent, although some manual intervention, particularly in the later stages, may be required.

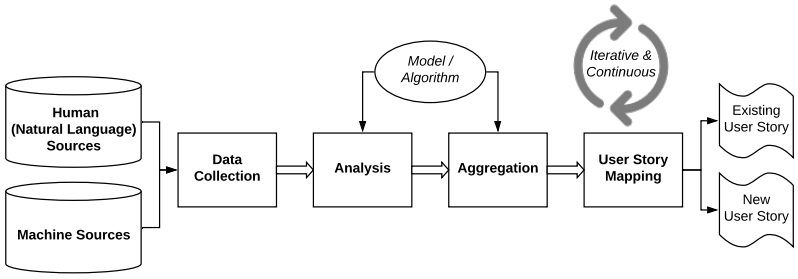

Figure 2. The process of managing the data in an instantiation of the meta-model.

Data Collection. One of the very first steps is to identify relevant data sources that should be monitored for requirements-relevant information. *DigitalSource* can be of various types and are in the proposed meta-model categorized according to whether the data is primarily generated by (i) humans (*NLSource*), and thus mainly in the form of a natural language, or (ii) machines (*MachineSource*), e.g. sensor data (*SensorReading*) or computer logs of various kinds (*ServerLogEntry*, *UserLogEntry*). *The NLSource* can come in many different shapes and forms, including *E-mail*, *Review*, *MicroblogPost*, *ForumPost*, *TrackedIssue* from an issue-tracking system, and *PolicyDocument*. These come with various meta-data and some may also have a *Measure* of a particular *MeasureType* that could provide requirements-relevant information; examples include the number of likes and retweets of a microblog post, or the review score of an app review. The particular language used in the free-text component (body) of an *NLSource* is highly relevant for the subsequent analysis of the data; often this information is not directly available as meta-data, but it is generally possible to apply a pre-trained language detection tool with high accuracy. The identified data sources are then continuously surveilled for requirements-relevant information.

Analysis. As data is collected, it is continuously analyzed in a fully automated fashion, either in (near) real-time or according to some specified time intervals. Fully automated analysis is critical as, in many cases, the data is generated at such high volumes and velocity that it would not be possible to analyze the data manually, resulting in the potentially valuable data being discarded. In other cases, a single source instance can be very rich in information and the automated analysis serves to identify the source instance as, indeed, relevant, as well as to extract relevant information in order to facilitate the work of the requirements engineer.

Different analytical tasks and methods are needed depending on the type of data. For *NLSource*, which is often rather unstructured, NLP is required to extract relevant information. Common analytical tasks for *NLSource*, and which have been included in the proposed meta-model, include: (i) classification, (ii) sentiment analysis, and (iii) NER. The outputs of these tasks are associated with a *Segment*, which allows for the body of an *NLSource* to be divided into smaller units, e.g. paragraphs or sentences, which can be analyzed separately. If one prefers to analyze an *NLSource* as a single

unit, it would be treated as a single *Segment*. For *MachineSource*, one is typically interested in identifying some form of *Behavior*. By analyzing *ComputerLogEntry*, one might, for instance, be able identify users attempting to perform a task in an unexpected and suboptimal manner. By analyzing *SensorReading*, one may discover statistical anomalies that, in turn, could provide insights regarding requirements.

Analysis of *DigitalSource* — whether in the form of *NLSource* or *MachineSource* — requires processing of the data. 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*, *Behavior*) 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 machine learning.

Aggregation. Once a set of *DigitalSource* — of potentially varying types — has been analyzed, i.e. *AnalyzedContent*, related data are grouped into an *Aggregation*, which may contain both *Behavior* and *Segment*-related analytical information. *Aggregation* can be achieved automatically or semi-automatically through, e.g., clustering and is specified by *Processing*. Data may, e.g., be aggregated around a specific functionality. An *Aggregation* is further analyzed and summarized as a *CandidateRequirement*, either by mapping to an existing one or creating a new one. Over time, many *Aggregation* instances may be linked to one *CandidateRequirement*.

User Story Mapping. Once a new *CandidateRequirement* is instantiated, an automatic mapping to existing *UserStory* instances is attempted. The attempted mapping result, along with information contained in the *CandidateRequirement*, is reviewed by the requirements engineer, who is also alerted when a *CandidateRequirement* is updated by a new *Aggregation*. The requirements engineer decides if the *CandidateRequirement* pertains to an existing *UserStory*, resulting in an update, e.g. *Functionality* refinement by *Tasks* or increased *Priority* for development in the *Backlog* due to additional "likes". The requirements engineer may also choose to create one or more new *UserStory* instances based on a *CandidateRequirement*, or to ignore it if insufficiently complete or deemed to be of too low importance.

3.3 Validation

The development of the meta-model engaged the authors of this study, as well as three reviewers: a senior academic, an experienced data architect from a global insurance company, and a SAFe- and Scrum-certificated expert from telecom services with considerable experience in different RE methods. The authors made a draft proposal for the meta-model and a process for its use, followed by exploratory interviews that were analyzed for correctness, completeness, and usability by the reviewers until a commonly agreed result was obtained. The feedback of the reviewers was of great benefit for (i) improving syntax and semantic of some classes and relationships, (ii) elaborating concepts and activities between the sources and the User Stories, and (iii) for discussing uses and benefits of the framework in the organizational context.

4 Portal for Research Funding Proposals

The intention of this section is to demonstrate how the proposed meta-model and the process can contribute to more holistic data-driven requirements elicitation in a business organization. The example is based on a custom-developed web application for managing research proposals at a university in Sweden. In essence, the application, referred to as ResearchPortal, is used to store research proposals, and to approve them from research quality, budget, regulative, and ethical perspectives. Like other custom-made solutions, due to changing needs of users, technology, and business policies, the application is subject to continuous evolution. After an initial release four years ago and following an agile approach, the main sources of requirements are currently emerging in a digital form, i.e. online reviews, e-mails, microblogs, and changes to business policies. Due to the magnitude and diversity of sources, an automated aid for collection, aggregation, and mapping to the existing requirements is highly needed. The presentation of the illustrative case follows the process described in Figure 2.

Data Collection. We have collected three digital sources of three different *NLSource* types, i.e. composed in natural language: an *Email*, a *Review* and a *MicroblogPost*. The first *DigitalSource* is an *Email* addressed to the product owner of ResearchPortal.

DigitalSource: NLSource: Email			
Body	Hi! Would it be possible to add founding organization as one of the search fields? That would help the management to find proposals. That would be great! Best regards, John	*Attachment*	N/A
		Language	English
		Subject	ResearchPortal, a new search keyword?
		Sender	john@university.com

The second *DigitalSource* is a *Review*, representing explicit feedback intended for requirements elicitation. The creator is anonymous and there is no associated *Measure*.

DigitalSource: NLSource: Review			
Body	I suggest we make it easier for researchers to find past proposals that they have participated in. It should be possible to search using multiple criteria.	*Language*	English
		Type	Suggestion Box
		Creator	Anonymous

The third *DigitalSource* is a *MicroblogPost* and, in this case (as there is no @ tag used to get the attention of the product owner), it can be seen as a form of implicit feedback and not intended for requirements elicitation. In this case, there are two measures associated with the *MicroblogPost*, one of *MeasureType* Likes and the other Retweets. During data collection, an initial filtering is based on the keyword "ResearchPortal".

DigitalSource: NLSource: MicroblogPost			
Body	It's so difficult to find proposals in ResearchPortal!	*MeasureType*	Likes
Language	English	*Measure*	10
Service	Twitter	*MeasureType*	Retweets
Sender	dr_emma	*Measure*	15

Analysis. Different types of analyses can be performed, both at different points in time and for different types of data. For *NLSource* data, we need to decide whether to split the body into multiple text segments (e.g. paragraphs or sentences) or to treat it as a single unit, i.e. a single segment. In this case, we have chosen to split the *Email* into sentences. The body would be split into five segments; in the table below, we have excluded "Hi!" (Segment 1) and "Best regards, John" (Segment 5). For both the *Review* and *MicroblogPost*, we have chosen to treat them each as a single unit.

Each segment is then analyzed separately. For each of the *NLSource* instances, we have carried out the same analytical tasks. However, note that we may need to apply different *Processing* and potentially different *Model* instances for different types of data to get the best results, due to differences in the distributions of the data that are difficult to capture well in a single model. For example, the language use in microblog posts versus emails can be rather dissimilar in terms of vocabulary and grammar. We analyze the data by applying classification, NER and sentiment analysis. A segment can hence be associated with one or more *Classification*: here, we apply one classifier for the classes Feature Request, Bug Report, and Other; as well as another classifier for the classes Functional Requirement, Non-Functional Requirement, and Other. Each segment can moreover be associated with one or more *NamedEntity*: here, we apply a NER model to identify the entities Functionality, Benefit, Role. Finally, each segment may also be associated with one or more sentiment, i.e. Positive, Negative or Neutral. The results of the analyses are shown in the tables below.

DigitalSource: NLSource: Email		
Segment 2: Would it be possible to add founding organization as one of the search fields?		
Classification	*Label*: Feature Request	*Label*: Functional Requirement
NamedEntity	*Content*: add founding organization as one of the search fields, *Type*: Functionality	
Sentiment	*Label*: Neutral	
Segment 3: That would help management to find proposals.		
Classification	*Label*: Other	
NamedEntity	*Content*: management, *Type*: Role	*Content*: find proposals, *Type*: Benefit
Sentiment	*Label*: Neutral	
Segment 4: That would be great!		
Classification	*Label*: Other	
NamedEntity	N/A	
Sentiment	*Label*: Positive	

Segment 2 is classified as a Feature Request and concerning a Functional Requirement, while a named entity in the form of a Functionality is identified. In Segment 3, two named entities are identified: a Role and a Benefit. The sentiment of Segment 2 and 3 is neutral, while there is a positive sentiment in Segment 4.

DigitalSource: NLSource: Review			
Segment 1: I suggest we make it easier for researchers to find past proposals that they have participated in. It should be possible to search using multiple criteria.			
Classification	*Label*: Feature Request	*Label*: Functional Requirement	
NamedEntity	*Content*: researchers, *Type:* Role	*Content*: find past proposals, *Type*: Benefit	*Content:* search using multiple criteria, *Type*: Functionality
Sentiment	*Label*: Neutral		

The review is classified as a Feature Request and concerning a Functional Requirement. Three named entities are identified: “researchers” as Role, “find past proposals” as Benefit, and “search using multiple criteria” as Functionality. The sentiment is neutral.

DigitalSource: NLSource: MicroblogPost		
Segment 1: It’s so difficult to find proposals in ResearchPortal!		
Classification	*Label*: Other	*Label*: Other
NamedEntity	*Content*: find proposals, *Type*: Benefit	
Sentiment	*Label*: Negative	

In the tweet, a named entity in the form of a Benefit is identified, while the sentiment is deemed negative by the sentiment analysis model. In this example, we do not process the measures associated with the tweet any further at this stage, but pass along the information to the later stages.

Aggregation. *Aggregation* allows for combining requirements information from one or more instances of *DigitalSource* and of different types collected over some time period. To facilitate this, *Segment* and *Behavior* are generalized into *AnalyzedContent*. The analyses associated with *Segment*, i.e. *Classification*, *NamedEntity* and *Sentiment*, are passed along but not further processed at this stage. An *Aggregation* simply links together one or more related *Segment* and/or *Behavior* over some time period.

In this example, all *Segment* instances become part of the same *Aggregation* as they all concern the need to find proposals in ResearchPortal. However, it would also be possible for multiple *Aggregation* instances to be created — this is decided by the aggregation algorithm or model. An *Aggregation* is then mapped to existing *CandidateRequirement* instances. If there is no match, a new *CandidateRequirement* instance is created; if there is a match, the existing *CandidateRequirement* is updated. At this stage, the *AnalyzedContent* instances are further processed and summarized such that the information can support the decision-making of the requirement engineer in the next stage. The manner in which the data is analyzed at this point is specified through *Processing* and associated classes.

User Story Mapping. Once there is a new *CandidateRequirement*, an automated mapping with existing *UserStory* instances is attempted. Again, the mapping procedure is specified through *Processing* and associated classes. The results of the attempted mapping are shown to the requirements engineer. If an existing *CandidateRequirement*

is updated with newly collected and analyzed data, the requirements engineer would similarly be notified.

At this stage, there is likely a need for manual intervention and the decision to create or update an existing *UserStory* rests with the requirements engineer. In this case, we assume there is no existing *UserStory* and that the requirements engineer creates one based on the collected and presented information within *CandidateRequirement*.

User Story: As a researcher, I should be able to search for a proposal by a funding organization so that I can easily obtain all my previous proposals to the funding organization.	
Role	Researcher
Functionality	Search for a proposal by funding organization
Benefit	Easily obtain all my previous proposals to the funding organization

Other quantitative measures from *CandidateRequirement* can provide useful information at this stage, e.g. for prioritization. The fact that the tweet complaining about the difficulty to find past proposals was liked and retweeted several times can be an indication of the urgency to improve this aspect of ResearchPortal. This may lead to a change of the *Priority* of the *UserStory* and therefore its placement in *Productlog* for development; the decision may be programmatic, and based on different algorithms, i.e. who asked for the change, how many, or from how many different sources; or the decision may be done manually, at the mapping stage.

5 Discussion

In this study, we have presented the foundations for a novel framework that integrates heterogeneous data sources with User Stories for the purpose of automated and continuous requirements elicitation and management. The main motivation lies in the fact that increasing digitalization and Big Data are creating opportunities for eliciting system requirements from new, digital sources; however, due to the scale with which data is generated, along with the heterogeneity of sources and associated challenges, it is not feasible to analyze the data manually. The data-driven framework is aimed to serve as a tool for organizations striving to manage requirements for evolution of their various software systems in a structured and holistic manner.

The primary focus of this paper has been to define the main meta-model, which aims to conceptually integrate heterogeneous digital data sources, the means for their processing, as well as to map the extracted and aggregated data to requirements artifacts. As such, it is aimed for storing the information and, by means of a supporting software tool, enable highly automated requirements management. However, as described in the paper, the process is best viewed as semi-automatic, requiring certain manual activities. This is in part due to the fact that User Stories are semi-informally

defined artifacts, as are their tasks and constraints. Other reasons stem from the challenges pertaining to processing heterogeneous and often rather unstructured data sources. There is still a need for further research concerning the application of ML and NLP to RE, in particular concerning integration of data from heterogeneous sources. From the validation, it is concluded that the meta-model is reasonably detailed; however, further elements could be added, especially regarding possible details concerning data processing. On the other hand, we have also aimed for flexibility in allowing for different choices to be made with respect to the analysis of data, as this is highly context-dependent.

The process of the framework is described on a high level of abstraction. It needs to be further elaborated towards a methodology for software evolution by taking into account a synergy of stakeholder-driven agile methods for requirements elicitation with the data-driven counterparts; we have found some related ideas in a relevant paper [24]. Some highly interesting aspects of the methodology concern elaboration of how "interactions with the customer" compare to traditional agile, as well as the frequency of requirements processing from organizational and project-based perspectives, keeping in mind that streaming data facilitates iterative requirements discovery to become increasingly continuous [25].

Future work includes (i) development of a prototype for implementing the proposed framework, (ii) further elaboration of NLP and ML techniques for classifying their use according to the specifics of User Stories and related elements, and (iii) proposals for "learning" solutions based on manually repeated activities and discovered patterns.

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 (2000)
2. Pohl, K. Requirements engineering: fundamentals, principles, and techniques. Springer, Heidelberg, New York (2010)
3. Rubin, K.S.: Essential Scrum: a Practical Guide to the Most Popular Agile Process. Addison-Wesley, Upper Saddle River (2012)
4. 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: Proc. of the Practice of Enterprise Modeling (PoEM), LNBIP vol.335, pp. 205-220, Springer (2018)
5. 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)
6. Dąbrowski, J., Letier, E., Perini, A., Susi, A.: Mining User Opinions to Support Requirement Engineering: An Empirical Study. In: Proc of int. Conference on Advanced Information Systems Engineering, LNCS, vol. 12127. pp. 401-416, Springer (2020)
7. Cohn, M.: User Stories Applied: for Agile Software Development. Addison Wesley, Redwood City (2004)
8. Zdravkovic, J., Svee, E-O., Giannoulis, C.: Capturing Consumer Preferences as Requirements for Software Product Lines. RE Journal vol. 20/1, pp. 71-90, Springer (2015)

9. Nguyen, V., Svee, E-O., Zdravkovic, J.: A Semi-Automated Method for Capturing Consumer Preferences for System Requirements. In: Proc. of The Practice of Enterprise Modeling (PoEM), LNBIP vol. 267, pp. 117-132, Springer (2016)
10. 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 (2013)
11. Manrique-Losada, B., Zapata-Jaramillo, C. M., Burgos, D. A.: Re-Expressing Business Processes Information from Corporate Documents into Controlled Language. Natural Language Processing and Information Systems, pp. 376–383, Springer (2016)
12. Nicolás, J. Toval, A.: On the Generation of Requirements Specifications from Software Engineering Models: A Systematic Literature Review. Information and Software Technology, vol. 51/9, pp. 1291–1307, Elsevier (2009)
13. 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)
14. Ahmed, M. A., Butt, W. H., Ahsan, I., Anwar, M. W., Latif, M., Azam, F.: A Novel Natural Language Processing (NLP) Approach to Automatically Generate Conceptual Class Model from Initial Software Requirements; In: Proc. of International Conference on Information Science and Applications (ICISA), LNEE, vol. 424, pp 476-484, Springer (2017)
15. Shao, F., Peng, R., Lai, H., Wang, B.: DRank: A Semi-Automated Requirements Prioritization Method Based on Preferences and Dependencies. Journal of Systems and Software, vol. 126, pp. 141–156, Elsevier (2017)
16. Dhinakaran, V. T., Pulle, R., Ajmeri, N., Murukannaiah, P. K.: App Review Analysis via Active Learning: Reducing Supervision Effort without Compromising Classification Accuracy. In: Proc. of 26th Int. Req. Eng. Conference (RE), pp. 170–181, IEEE (2018)
17. Williams, G., Mahmoud, A.: Mining Twitter Feeds for Software User Requirements. In Proc. of 25th Int. Requirements Engineering Conference (RE), pp. 1–10 IEEE (2017)
18. Xiao, M., Yin, G., Wang, T., Yang, C., Chen, M.: Requirement Acquisition from Social Q&A Sites. In Proc. of 2nd Asia Pacific Symposium (APRES) , vol. 558, pp 64-74 (2015)
19. Morales-Ramirez, I., Kifetew, F. M. Perini, A.: Analysis of Online Discussions in Support of Requirements Discovery. In: Proc. of Int. Conference on Advanced Information Systems Engineering (CAiSE), vol. 10253 LNCS, pp. 159–174, Springer (2017)
20. Xie, H., Yang, J., Chang, C. K., Liu, L.: A Statistical Analysis Approach to Predict User's Changing Requirements for Software Service Evolution," Journal of Systems and Software, vol. 132, pp. 147–164, Elsevier (2017)
21. 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. Journal of Manufacturing Science and Engineering, vol. 141, p. 021010, ASME (2019)
22. Maalej, W., Kurtanović, Z., Nabil, H., Stanik, C.: On the Automatic Classification of App Reviews. Requirements Engineering Journal, vol. 21/3, pp. 311–33, Springer (2016)
23. Henriksson, A.: Learning Multiple Distributed Prototypes of Semantic Categories for Named Entity Recognition. International Journal of Data Mining and Bioinformatics, vol. 13, no. 4, pp. 395-411 (2015)
24. 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 Int. Conference on Advanced IS Engineering (CAiSE), LNCS, vol.10816, pp.603-618, Springer (2018)
25. Kirikova, M.: Continuous Requirements Engineering. In: Proc. of International Conference on Computer Systems and Technologies (CompSysTech), pp 1–10, ACM DL (2017)