Archetypes of delay: An analysis of online developer conversations on delayed work items in IBM Jazz

Abdoul-Djawadou Salaou, Daniela Damian, Casper Lassenius, Dragoş Voda, Pierre Gançarski

Version of Record: https://www.sciencedirect.com/science/article/pii/S0950584920301907
Manuscript_7d61acaad17fde6de39fb6faa0e107fd

Archetypes of Delay: An Analysis of Online Developer Conversations on Delayed Work Items in IBM Jazz

Abdoul-Djawadou Salaou[a,b], Daniela Damian[a], Casper Lassenius[c,d], Dragos Voda[c], Pierre Gançarski[b]

[a] *University of Victoria, SEGAL Lab*
[b] *University of Strasbourg, ICube Lab*
[c] *Aalto University, Espoo, Finland*
[d] *Simula Metropolitan Center for Digital Engineering, Oslo, Norway*

Abstract

Context. A widely adopted methodology, agile software development provides enhanced flexibility to actively adjust a project scope. In agile teams, particularly in distributed environment, developers interact, manage requirements knowledge, and coordinate primarily in online collaboration tools. Developer conversations become invaluable sources to track and understand developers' interactions around implementation of requirements, as well as the progress of implementation relative to the project scope and the planned iterations in agile projects. Although extensive research around iteration planning exists, there is a lack of research that leverages developer conversation data to understand delays in implementing planned requirements in agile projects.

Objective. By using developer conversations in a large agile project at IBM, this work aims to analyze conversation in work items (WIs) that are delayed and derive patterns that suggest reasons for delay in the project.

Method. We conducted a case study of the IBM Jazz project, and used thematic analysis to code the developer conversations as time-series, and cluster analysis to identify patterns that differentiated the evolution of discussions in WIs that were late vs. not late in the project.

Results. We identified six main patterns of WI delay. Through semantic analysis of developer conversations within particular clusters we were able to explain the reasons for delays in each pattern. In comparison to non-late WIs, we find that the major reason for delay is a lack of frequent communication associated with a poor project management of WIs. Similarly, non-late tasks more often

Email addresses: adsalaou@unistra.fr (Abdoul-Djawadou Salaou), Daniela.damian@uvic.ca (Daniela Damian), casper.lassenius@aalto.fi (Casper Lassenius), dragos.voda@aalto.fi (Dragos Voda), gancarski@unistra.fr (Pierre Gançarski)

delegate to children tasks to accelerate the implementation of requirements, in addition to processing requests quickly to resolve bottlenecks in implementation.

Conclusion. Our study complements existing research in bringing evidence that developer conversations are a useful resource that can highlight delays in requirement implementation, as well as recommend patterns in the dynamics of developers interactions relevant to such delays.

Keywords:
Text analysis, Software engineering, Agile development, Categorical time series, Clustering, Thematic analysis, Task completion, Iteration completion, Repository mining, Jazz repository

1. Introduction

Contemporary software development is increasingly conducted by distributed teams using agile software development methods. These methods rely on a core set of principles and values, most importantly the capability to efficiently react to change [5]. This capability is often implemented through the use of short development iterations and a strictly prioritized list of requirements, which in turn are broken down into work items small enough to be implemented in a single iteration [58]. Agile methods have been shown to provide benefits both for small and large projects with respect, e.g., to on-time delivery and customer satisfaction .

Despite their success, it is not uncommon for work planned for an iteration to be deferred to later ones. Indeed it is often even recommended to plan for more work than can be done in order to avoid downtime within agile teams. The soundness of this approach has been debated, but regardless, empirical evidence points to the fact that decisions with respect to delaying work to later iterations are routinely made even in agile software development [27].

In addition to the question of whether or not to plan for more work than realistically can be accomplished in an iteration, another important question emerges: Are there archetypal situations regarding work items that can predict or explain the need for delaying them to later iterations?

In this paper, we present results from an empirical analysis of work items that were delayed, i.e., not delivered in the iteration they initially were planned for. Our research was guided by the research questions:

RQ 1 Can developer discussion threads on late work items be used to understand the reasons for delay?

RQ 2 What can we learn from comparing the discussions as well as other properties of the late and non-late work items?

While previous studies have leveraged various work item meta-data to predict a work item implementation timeline (e.g. [15]), in this paper we focus on

the developers' conversations to understand why work items get delayed in software projects and miss their planned iterations. Research already demonstrated that developers communication is useful in identifying patterns that might indicate problematic requirements [35]. We use thematic analysis on developer conversation data in combination with time-series and cluster analysis to identify discussion topics in developer conversations and subsequently reasons for delays in work item implementation. As an analysis method, time series analysis provides a unique opportunity to study the temporal evolution of discussions related to a work item in our data set. By coding the sequence of comments in developers conversations into time series, we studied the temporal distance and the order of these comments in an effort to characterize the discussions associated with late work items. Time series analysis has been recognized as providing a global view of the data by highlighting the temporal relationship between data points, and thus helping identify cyclic patterns, overall trends, turning points and outliers [47, 19].

Our findings show that late work-item tasks exhibit different patterns of delay. We identified six patterns with a consistent semantic meaning. For example, we find that communication and management problems are the common reason for delays in work-items implementation. The paper is structured in the following way: next, we introduce related work, followed by our research methodology. Then we present our results, focusing in the identified thematic clusters. Finally, we discuss the implications and limitations of the study.

2. Related work

Most research that has studied delayed work in software engineering has been done in the area of release and iteration planning. From a Release Planning perspective, Zowghi et al. [17] compared several prediction techniques for classifying the readiness of an ongoing release, while Deghan et al. [15] employed a process of feature engineering and machine learning to predict the likelihood of feature implementation within a planned iteration. In the same vein of research around delivery capabilities and with the goal of assisting project managers, Choetkiertikul et al. leverage historical data for creating a predictive model that can forecast the amount of work delivered compared to the initial commitment [10]. In a different study, the same research group present a model that predicts both the degree of delay and the likelihood of the delay occurrence, for a software project issue against its due date [9].

While, to our knowledge, no other studies investigated developer communication to understand and explain delayed work items, communication threads by developers have been previously studied. For instance, Knauss and colleagues [35] propose a method that analyzes requirements communication data in order to timely detect and raise awareness of requirement related risks during implementation. Licorish and MacDonell [38] studied the attitudes of teams members extracted from discussion threads, and how they relate to task performance, while Kavaler et al. [32] looked at language complexity levels and

how they affect issue resolution time. Yilmaz [69] tries to capture the software teams personality traits impact on software development by conducting context-specific survey.

In addition to the above, there are several studies that have used Social Network Analysis to study developer communication, e.g., [67], [7], [44], [14]. However, such analyses ignores the actual message content and dynamics. In this study we used time series and clustering methods to model and group work item discussions that present certain similarities that might indicate reasons for delay in their implementation. In software engineering research, time series analysis modeling has been used in approaches to study software reliability [2, 1] and to identify temporal patterns of software evolution defects [52]; while clustering techniques proven useful in analyzing similarities in software measurement data to distinguish between fault and non-fault prone software modules [71] or to identify outliers in such data [70].

3. Research Setting and Methodology

We conducted a case study of developer online conversations during the planning and implementation of Jazz software components modules using the Jazz collaboration platform of IBM[1]. As a product, Jazz has been operational since 2006 and functions as the base platform for many of IBM's services such as Rational Team Concert or Rational Quality Manager. It aims to improve software practices, collaborative work and management processes by creating a scalable platform which can coordinate tasks and provide improved visibility throughout the software development life cycle [53].

Because Jazz is an integrated development environment, it includes supporting tools for planning, software builds, code analysis, version control as well as online communication, allowing developers to use the same tools for development and coordination. This approach grants access to a wealth of rich data concerning software development characteristics as well as communication and collaboration data, gathered in a timely and non-invasive manner, as compared to conducting surveys or interviews.

The teams developing Jazz use the Eclipse way methodology [22], similar to the Open Unified Process and partly conforming to agile principles, that defines iteration cycles between two to six weeks consisting of three stages, namely planning, development and stabilization. Longer iterations, varying in length from one month to a year, exist as release iterations and contain multiple, shorter, milestone iterations that each end with a new milestone build. The goals and features for each release are defined by project management prior to the start of the iteration and captured in work items as task descriptions. Development is conducted through these work items and are assigned to a release or a milestone iteration but can be postponed in case of delays.

[1] https://jazz.net

This monitoring of the real work environment aims to capture all the critical information and discussion generated by the developers and offer an overview of the project and its evolution, with an ability to go back and analyze certain events if needed.

3.1. Work Items and Developer Discussions

In our data set, a work item (WI) describes a unit of work representing a singular assignable task [15]. Table 1 display the major attributes describing a work item. There are different kinds of work items, and they form a hierarchy (see Table 2); from *Plan Item* (top level) to *Task* (low level) such that a type can have a sub-type as children [15]. However, in practice, it is possible that this recommended order is disregarded with children sharing the same type as their parent or Tasks connected directly to a Plan Item. A Defect can be a child of any other type.

Table 1: Overview of Work Item descriptive attributes

WI attribute	**Possible values**	**Detail in**
Type	Plan Item, Story, Enhancement , Defect, Task	Section 3.1
Description	Textual description of the WI goals	
Creation date	WI open date	
Panned For	Iteration Id	
Estimate	Implementation duration estimation	
Resolution Date	WI close date	
Status	On-Track, Behind, Suspended, Abandoned, Done	
Discussion	Conversation around WI implementation	Section 3.1.1
Severity	Unclassified, Minor, Normal , Major, Critical, Blocker	Section 3.1.2
Priority	Unassigned, Low, Medium, High	
Created By	@UserX	Section 3.1.3
Owned By	@UserY - WI responsible / coordinator	
Subscribed By	Collaborators user list	
Resolved By	@UserY	

Table 2: Overview of Work Item types

Work Item Type	**Explanation**
Plan Item	Top level items representing requirements or features to be included in future releases.
Story	High level items dividing the work from Plan Items into subsequent iterations.
Enhancement	Item that adds functionality or extends existing features.
Task	Detailed item contained within a single iteration.
Defect	Item representing work required for bug fixing.

Each WI contains information related to its parent, children, priority, deadline, conversation history, and the person responsible for it. During a development phase, a WI cannot be marked as completed until all its children tasks are

completed. We consider a late WI to be any WI that is still under development past its currently assigned iteration end date. If a WI is still under development while its assigned iteration ends, but is postponed, either before or after the deadline to a future iteration, it is not considered late. An example of possible WI evolution can be seen in Figure 1.

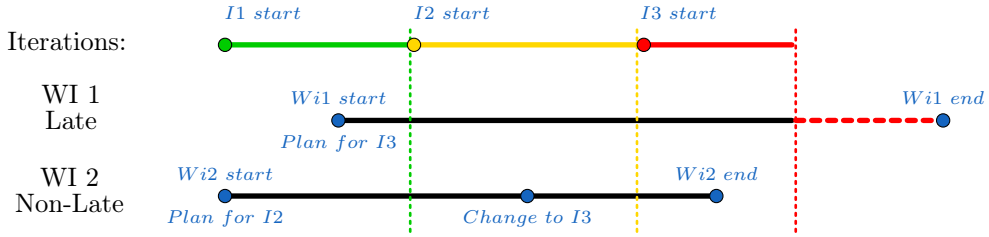

Figure 1: Late and non-late work-item examples

3.1.1. Dataset Description

Our research dataset included work items and their conversations between 2011 and 2016, over a total of 612 iterations. We analyzed the discussions for all completed work items of all types[2] that had at least fifteen comments in their discussion threads. We chose fifteen comments as it is a reasonable number to obtain insights on the progression of work on the work items. Since we are focusing mainly on comments data, discussion with very few comments are not significant to highlight all the development stages (elicitation, implementation, clarification,..) in a work item's lifecycle. We also ruled out WIs with a particular status - *invalid, abandoned, suspended* - that indicated the work item did not follow the normal development process. This filtering resulted in a total of 125 late work items and a total of 2655 associated comments. To analyze the differences and similarities between late and non-late work items we also analyzed the discussions associated with the same number (125) of randomly selected non-late WIs, a total of 4068 comments. For each of these WIs, we extracted all associated comments in its related conversation, preserving their order, to create what we refer to as "WI discussion". Table 3 provides an example of a work item discussion.

3.1.2. Dataset Overview

The *Estimate* attribute (Table 1) gives the estimated duration required for WI implementation. This information is not frequently provided in our dataset. Only 26% (32) of non-late and 16% (20) of late WIs have this information. *Severity* describes the importance of a WI and Priority its implementation emergency level. Depending on these attributes values, a sensitive WI is likely to get more attention and resource for its implementation speedup. However, in our dataset,

[2]With the exception of Tasks, which did not contain any implementation comments

Table 3: Example of Work Item discussion

Work ID	Comment Number	Comment Text
115331	1	Extracted from work item 113204.
	2	Additionally, we could implement a substitution parameter in the error string so it could read something like: Your client is not compatible with the server...
	3	Running foundation.stable.jcb RJF-T20100625-0850 with these changes and then I will try writing the client half to make use of this. re comment 2: Why does the server return the message? The client just needs to know that it's a mismatch and what the server version is. The client could do the message formatting on it's own side which will be in the locale/language of the client
	4	The idea is that a server admin can update a setting on the server and provide a custom message to all backlevel clients, perhaps providing a link to an internal website with client upgrade instructions.
	5	@user I have completed 2 changesets here for this. It sends the server version and puts up a new dialog with the link. Please review and deliver this if you like it.

a comparison of these attributes distributions between late and non-late WIs (Tables 4 and 5) highlights no significant differences, i.e. both sets have more or less the same distributions. Therefore, they can be discarded as the attributes that discriminate late WIs from non-late ones.

Table 4: WI severity distribution

Severity	#Non-late	#Late
Unclassified	33 (26%)	47 (37%)
Minor	3 (2%)	3 (2%)
Normal	56 (45%)	52 (42%)
Major	22 (18%)	17 (14%)
Critical	6 (5%)	4 (3%)
Blocker	5 (4%)	2 (2%)

Table 5: WI priority distribution

Priority	#Non-late	#Late
Unassigned	31 (25%)	35 (28%)
Low	2 (2%)	0 (0%)
Medium	20 (16%)	32 (26%)
High	72 (57%)	58 (46%)

3.1.3. Work Items Management

The *Created By* attribute (Table 1) identifies the user who created the WI and provides information related to its goals, while the *Owned By* identifies the user who is going to drive or coordinate the WI implementation to resolution. The *Resolved By* identified the user who closed WI as resolved. This attribute has always the same value as the owner in our dataset, for instance the owner closes the WI when it is implemented. The *Subscribed By* attribute holds a list of collaborating users to implement WI requirements.

Plan Items do not have a creator. They only have owners. Because a plan item is a high level description of a feature, its implementation involves the collaboration of teams working on dependent work items such as Stories, Enhancements, Defects or Tasks. A development "team leader" coordinates work

of a team of developers to deliver its assigned requirements. Further, there are Project Leaders who lead the work at each geographical location.

With respect to the workflow around a WI implementation, typically a WI is delegated to the appropriate developers, i.e., the creator of WI is different from the owner (@UserX != @UserY). For example when a @UserX spots a defect and creates a WI to report it and assigns it to @UserY (more capable to handle the WI); or when a Plan Item is decomposed into stories and assigned to the appropriate developers/team. Some WIs in our dataset indicate self-management, i.e. the owner of the WI is also the creator (@UserX = @UserY), and it is typical of WIs of the type Enhancements. In our dataset 36% of non-late and 27% of late WIs are in this situation (Table 6).

Table 6: WI management distribution

WI management type	**#Non-late WIs**	**#Late WIs**
Self (Creator is the Owner of the WI)	45 (36%)	31 (27%)
Delegate (Creator is not the Owner of the WI)	80 (64%)	94 (73%)

3.2. Methodological Steps

To characterize the WI discussions, we sought to analyze clusters of WIs for which the associated discussions showed similarities in how they evolved over time (section 3.2.4). To this end, we used cluster analysis techniques [47, 55, 24] on WI discussions (section 3.2.3) that were analyzed and processed in a way to allow for such type of analysis.

To develop the elements in our cluster analysis (i.e. clusters of WI discussions and their items) we analyzed and processed the WI discussions as time series of comments that captured the temporal evolution of the discussions (section 3.2.2). The time series were developed by coding the WI discussions and assigning codes to groups of comments in the discussion; a code represents a label indicating the topic that best characterized the respective group of comments (section 3.2.1). This analysis was conducted on both the late and non-late WIs. Throughout this analysis we interacted closely with an IBM development team leader involved in the Jazz project during the iterations in our dataset to validate our understanding of the WIs, their context of implementation and details of the WI discussions.

To answer RQ1, we first analyzed the late WIs clusters and searched for patterns that might explain reasons for delays in the WI implementation. For RQ2, we then contrasted the properties of the late WI with that of the non-late WI clusters to further our understanding of reasons for delay in WI implementation.

We illustrate this staged analysis process in Fig. 2 and explain it in detail in the following subsections.

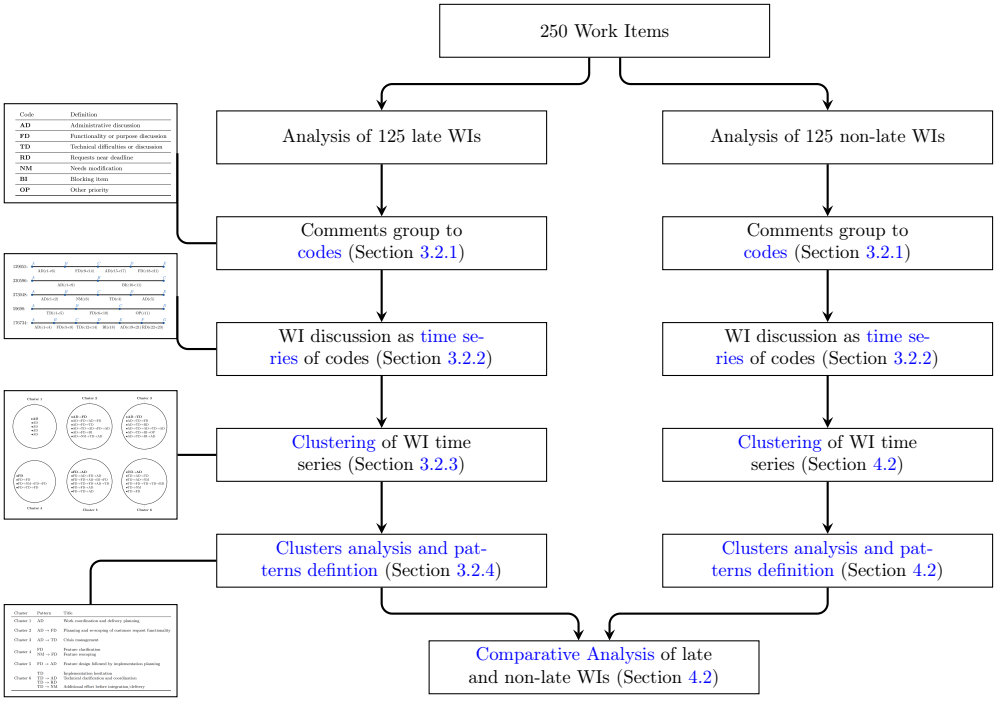

Figure 2: Research Methodological Steps

3.2.1. Thematic Analysis of Work Item Discussions

To characterize the WI discussion threads in a way that allowed for time series and cluster analysis, we used thematic analysis and developed codes that represented the main discussion topics of the discussion threads. Thematic analysis is commonly used in qualitative research. It is a systematic coding and categorization approach used for exploring textual data to identify themes, analyze them, and provide an explanation of the underlying patterns [8, 61]. While thematic analysis are widely used in social science [59] and health [16, 21], the software engineering literature contains only few uses of this method to analyze and characterize the software development process [12].

In thematic analysis, the most sensitive stage is the coding of the content under analysis, during which the data is categorized, manually or automatically, to facilitate analysis [57, 8]. Hay [26], recommends a two-step process beginning with basic coding in order to distinguish overall topics, followed by a more in depth, interpretive code in which more specific trends and patterns can be interpreted. While qualitative coding often is done manually by the researchers, contemporary natural language processing tools such as Weka [25] can be used to automatically categorize pieces of text into predefined categories. Such tools can help speed up coding processes that use a single pieces of text as an objects

for topic identification [40, 51]. This is the case for a priori -based coding where categories are available or can be easily deducted, as can be the case for, e.g., survey or interview transcript data [31].

In our analysis, we applied a conventional content analysis technique [28] to examine, review the conversations, and identify the main discussion topics as codes. A code in our approach was associated to a group of consecutive comments that were semantically related. Our approach was to shift the focus of analysis from the individual phrases and words in comments to the semantics, understood as core messages within groups of comments in the discussion. For example, we used the code AD (Administrative Discussion) to categorize sets of consecutive comments around planning of work and status updates regarding implementation, testing or delivery. Similarly, discussions related to needed functionality or its purpose within the project were given the code FD (Functionality or purpose discussion).

Two of the authors iteratively conducted the content analysis to increase the confidence in the identified codes. We achieved an inter-rater agreement of 83% using Cohen's kappa [36]. We validated our codes and their mapping to WI discussion comments groups in a series of meetings with a development team leader from IBM. We include the codes and their description in Table 7.

3.2.2. Time Series Analysis of Work Item Discussions

Once discussion codes were identified, we treated each WI discussion as a time series of the codes appearing in it, for an example see Figure 3. A time series represents a sequence of values obtained from sequential measurements over time. For instance, in our analysis WI 129055 is a sequence of four codes (AD → FD → AD → FD) where comments one to six (c1-c6) are identified as an AD code (Administrative Discussion).

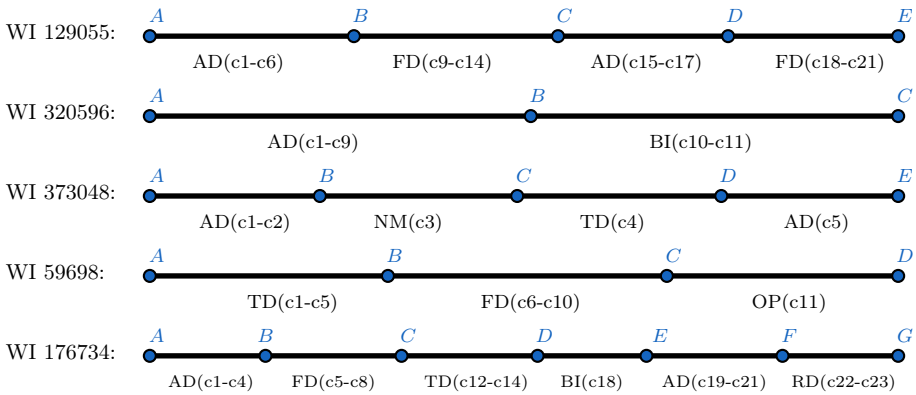

Figure 3: WI time series examples; the codes (e.g. AD, FD) characterize groups of consecutive comments (e.g. c1-c9)

Time series analysis comprises methods for analyzing time series data in

Table 7: Code definition and examples of groups of comments they characterize

Code	Definition	Example group of comments
AD: Administrative discussion	Discussion centered around planning of work as well as status updates regarding implementation, testing or delivery.	– *Finished implementing [...] plugin to use Composite Context UI (173302) to develop test UI for testing access groups, access group picker and composite contexts.* – *I updated the description of the parent plan item yesterday with my current understanding of what we should go after in 2012: [...].*
FD: Functionality or purpose discussion	Discussion related to what functionality needs to be implemented or its purpose within the project.	– *I don't feel strongly about this, but I'm wondering if the confirmation dialog should have OK/Cancel buttons rather than 'Update Stream'/Cancel buttons.* – *I'm a bit confused about what you are asking for.. Can you tell me how this would be different from the LDAP step in the setup wizard (/setup)?*
TD: Technical difficulties or discussion	Discussions of technical nature focused on solving the problems encountered during functionality implementation.	– *This new method caused RQM defect [...]. However, we can't implement this new method on the public [...] interface since the return type [...] is internal and it's package is not exported in the plug-in's manifest.* – *[Logs, Error messages, Code snippets, Component names]*
RD: Requests near deadline	One or more comments that prompt extra work near the WI deadline.	– *RRC will need to both migrate an existing project operation to a team operation. We also want to add new team operations to the default role.* – *Could you pls provide the proper steps to do that and how to validate the result so we can include this in our internal testing?*
NM: Needs modification	Single or multiple comments signaling required modifications before the implementation can be accepted.	– *Please make the following edits to [...] : [List of edits]* – *Some further comments: Change [...] to [...] and remove [...]*
BI: Blocking item	Discussions related to identified dependencies with other items, that are either in progress or not yet assigned, without which progress cannot continue.	– *Putting [current item] into backlog until a decision is made at the CLM PMC.* – *Deb is currently working the dependency relationship with the WAS team before we can submit*
OP: Other priority	Explicit statement that other WI demand more attention at the present time.	– *The open migration issues need to take priority over this. While this is a a bvt_blocker its not a critical path and the BVT team has worked around this.* – *Right now, this work item is number 6 down the priority list.*

order to extract meaningful statistics and other characteristics of the data [11]. Major time series related tasks include pattern recognition [29], clustering [55], classification [37], segmentation [33], query by content [20], anomaly detection [66], and prediction [63]. Categorical time series is a time series with nominal values. They have received increasing interest during the last years, and are used to mine qualitative sequential data in diverse fields of practice such as speech recognition [45], part-of-speech tagging [62, 34], biological sequence analysis [4, 18, 54], network monitoring [65] for intrusion detection [68], and many more. In this paper, we consider a WI conversation thread as a categorical time series with respect to the topics that appear along the thread. We provide all time series that we identified on a GitHub[3] repository and discuss their grouping in clusters in the following subsections.

We remark that in our analysis these are *categorical* time series. Unlike typical time series where the codes are numerical [19], the categorical time series represent a sequence of categories related temporally to one another [43, 54].

3.2.3. Time Series Clustering

To identify work items that had similar discussions, we used cluster analysis to group the work item time series. Clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar to each other than to those in other groups [56, 39]. The groups are called clusters and are formed by all time series that have *structural* similarity. The object at the center of cluster, the *centroid*, minimizes the sum of squared Euclidean distances between itself and each object in the cluster. The centroid can be thought of as the average or the representative object of the cluster [39]. With clustering, we can identify and summarize interesting patterns and correlations in the underlying data [24].

Our approach was to analyze the clusters of *late* work items and the codes within their associated time series. Our expectation was that within clusters of similar evolving work item discussions, certain conversation structures could be identified and abstracted as the defining property of the cluster. This property could then be later analyzed and associated with the cause of the delay in the work item.

For the process of clustering the WI time series, we used the k-means algorithm [3] which requires a metric to asses the similarity of two time series. To this end, we used Dynamic Time Warping (DTW) [49], which measures the similarity between two time series even if they do not have the same time span, meaning the time series do not have the same length [6]. Furthermore, because we are working with categorical data, a similarity matrix expressing the degree to which time series component items are similar or dissimilar to each other was needed to properly carry out the comparison. However, when combined with k-means clustering, DTW presents a unique challenge due to the averaging used

[3] https://github.com/salaouab/archtype-of-delay

in the calculation of cluster centers [30, 46]. To prevent the issue, we used the averaging method developed by Petitjean et al.(2011) that resolves the time series averaging problem [49].

3.2.4. Semantic Analysis of Work Item Clusters

The clustering approach provided a grouping of the WI discussions based on *structural* characteristics of the derived time series. To characterize the meaning of WI discussions within each cluster, we conducted a semantic analysis of comments within each time series. We sought to identify patterns in how the discussion progressed across the WIs in a particular cluster; our purpose was to check if members of a cluster share the same properties, and infer reasons for which WI implementation was delayed or not.

Two of the co-authors read all WI discussions in our data set and considered, for each WI, the following: 1) creation and resolution dates, 2) number and timestamp of comments in the discussion, 3) number of children WIs created during the discussion and which would indicate that implementation work started as the result of discussions, and 4) progression of codes in the WI time series. The codes, indicative of topics of discussion, and their sequence was particularly important for this analysis. The order of codes as well as the temporal distance between them gave an indication of how the topics evolved in the WI discussion. For example, a pattern of comments about planning the WI implementation (coded AD) that dominate the entire discussion until the very last comments that might be coded as TD (technical solution discussion) and no children WI are created, indicate a problematic WI that likely resulted in delay of its implementation due to lack of implementation activity early enough in the iteration. Table 8 outlines the clusters we identified and which we will discuss in section 4.1. These semantics and patterns we validated with the IBM development team leader over four one-hour long meetings during which we ensured that our understanding of the meaning in the WI discussions was correct and consistent with the development context in this IBM project.

4. Results

This section reports the semantic analysis of clusters as well as a comparative view of late and non-late WI.

4.1. Analysis of late work items

The codes we identified in our content analysis of WI discussions, together with their definitions and examples are listed in Table 7.

These codes allowed us to treat each WI discussion as a time series, and Figure 3 shows examples of such time series. The clusters we identified as groupings of similar time series are shown in Table 8, which also lists example WIs within their respective cluster. The first item shown in each cluster is the cluster centroid. The WIs were chosen so as to show that certain sequences

Cluster	WI#	#Com.	Time series				
Cluster 1	**395124**	**16**	**AD**				
10 time series	169941	20	AD				
	170833	15	AD				
Cluster 2	**220330**	**18**	**AD**	**FD**			
18 time series	350131	16	AD	FD	AD	FD	
	129055	22	AD	FD	AD	FD	
	356527	28	AD	FD	TD		
	72491	15	AD	FD	BI		
	313289	63	AD	FD	AD	AF	TD
Cluster 3	**360792**	**15**	**AD**	**TD**			
20 time series	187151	10	AD	TD			
	267693	15	AD	AD	TD		
	397883	15	AD	TD	RD		
Cluster 4	**106935**	**20**	**FD**				
14 time series	174513	23	FD	FD			
	361236	74	FD	NM	FD	FD	
	52997	15	NM	FD			
Cluster 5	**193729**	**17**	**FD**	**AD**			
34 time series	356023	54	FD	FD	AD		
	93154	27	FD	AD	FD	AD	
	73341	30	FD	FD	AD	BI	FD
	132199	39	FD	TD	FD	AD	TD
Cluster 6	**235967**	**30**	**TD**	**AD**			
30 time series	95301	22	TD	AD	TD		
	108122	24	TD	AD	NM		
	90783	18	TD	TD			
	161884	15	TD	TD			
	322783	34	TD	RD			
	254398	15	TD	RD			
	162657	16	TD	NM			
	341921	48	TD	NM			

Table 8: Clusters of Late work items, and example cluster members (chosen randomly); also included for each WI discussion: respective number of comments and time series. The bold cluster member is the cluster centroid.

Table 9: Late WI Clusters Overview

Cluster	Structural pattern	Semantic Pattern	Reason for delay
Cluster 1 10 time series	AD	Endless work coordination and delivery planning	Coordination and planning take too long to come to an agreement
Cluster 2 18 time series	AD → FD	Ongoing planning and re-scoping of customer request functionality	Prolonged discussion until an agreement on how to proceed is reached leading to the implementation process passing the planned iteration
Cluster 3 20 time series	AD → TD	Crisis management	Inadequately elicited requirement
Cluster 4 14 time series	FD	Endless feature clarification	Extended functional clarifications without timely decisions
	NM → FD	Feature rescoping resulting in work beyond current iteration	Major modification requirement
Cluster 5 34 time series	FD → AD	Feature design followed by implementation planning	Implementation requires additional work in children WIs that are being planned for future iterations
Cluster 6 30 time series	TD	Implementation hesitation	Extended technical clarifications
	TD → AD	Technical clarification and coordination	Additional planning and coordination required
	TD → RD TD → NM	Additional effort before integration delivery	No time left for implementation review and modifications

although apparently different, fall under the same pattern according to our analysis.

Table 9 outlines the six clusters we identified and the patterns inside each one; these patterns represent unique characteristics of groupings of WI discussions inside the cluster. Note that clusters can have more than one semantic pattern (e.g. Clusters 4 and 6). We describe each pattern and the reasons for delay of the WI implementation in the following subsections.

4.1.1. Cluster 1: Endless work coordination and delivery planning

Main Pattern: AD

Reason for delay: Coordination and planning take too long to come to an agreement.

The discussions in this cluster focus on work item delivery planning or brief work status updates, as shown in the two examples below.

> *I propose we target 7.0.32 at a minimum for 4.0.2. Also, we need to make this happen early in M2.*
>
> *I have implemented all of the changes [...]. I am working on a patch based on [...], and will test it on [...]. The implementation does not work.*

Often a single person keeps the team up to date: *I've finished an implementation and sent off a patch zip to others for independent testing.*

The comments do not indicate discussion around the actual WI implementation and the majority of WIs in this cluster have no associated child WIs, that might have otherwise indicated that work is being delivered in relation to these WIs.

Examining the timestamps of the comments, it appears that comments indicating any agreement on the WI are made only towards the end of the iteration and not sufficient time remains to finalize the WI implementation. Table 10 shows a typical discussion of this type.

Table 10: Example WI discussion (WI 244762) for pattern AD, Cluster 1

User	Day Gap	Comment	Theme
user1	0	@user2 - tomcat 7.0.32 was made available on 12/4 and contains additional fixes. I propose we target 7.0.32 at a minimum for 4.0.2. Also, we need to make this happen early in M2. Can we get this assigned an owner and target?	AD
user1	0	correction, to comment 1, 7.0.32 was available 10/9 but the latest security info was made available 12/4.	
user2	2	"Changing the title to reflect 7.0.32. I don't see that version approved yet thru OSSC so need @user3 and Bill Spurlin's help on this. I will be the plan item owner but Christopher Maguire will be the driver from Releng. Plan is to get this early in M2 https://w3.tap.ibm.com/w3ki07/display/OSSCProcess/All+ Packages+ List"	
user3	2	The WAS team is telling us that we should be using WAS Liberty Profile instead.	
user2	3	OSSC approval received for Tomcat 7.0.32	
user4	3	@user7 FYI	
user4	0	@user6 and @user5 what is the plan to update Tomcat in 4.0.0.2 and 4.0.1.1 to 7.0.32?	
user5	0	"@user4 There is no such plan that I know of. Since the 4.0.0.2 is in RC 2 next week. Updating Tomcat this late in the game for 4.0.0.2 does not seem a good idea."	
user4	0	@user5 I understand. I didn't get keyed into this problem until this week. Should it be planned for early 4.0.0.3 if that release is planned?	
user2	1	We should probably consider having separate tasks linked to this plan item for each release where we are going to be bundling Tomcat 7.0.32. It needs to go into 4.0.2 and 4.0.1.1.	

4.1.2. Cluster 2: Ongoing planning and re-scoping of customer requested functionality

Main Pattern: AD → FD

Reason for delay: Prolonged discussion until an agreement on how to proceed is reached leading to the implementation process passing the planned iteration.

Most of the work items in this cluster are related to customer feature requests

for software customization. An example discussion for a WI in this cluster is shown in full in Table 11.

For these WIs, the planning is, for the most part, about assigning an iteration in which the solution needs to be delivered in order to meet the customer expectations. The first few comments are typically setting a deadline (iteration or due date) for the feature implementation, as the following three examples illustrate:

> *Please also how much time you will need for each segment? I am looking to host this enablement the third week in June please be on the look out for the dates.*
>
> *[...] @userX, maybe you could comment on whether or not this can be done for iFix.1. Thanks!*
>
> *@userX can we do it for 3.0.0.1.*

The discussion for planning usually takes less than five comments to come to an agreement which is then followed by a discussion on feature characteristics.

Functionality discussion occupy most of the WI discussion in this cluster and it is mainly about scrutinizing the demand of the customer in the context of the development environment by evaluating the solution in terms of implementation speed and efficient integration, without any breaks and retro-compatibility issues:

> *[...] Bumping this up to Major severity (my customer actually has it classified as Critical in their ranking).*

Comments indicating an agreement on how to proceed are often among the last and towards the end of the current iteration, and any agreed implementation is often carried out outside the current iteration (Table 11). Another distinguishing property of this pattern is that there are no children tasks created to parallelize and speedup the WI implementation.

Table 11: Example WI discussion (WI 220330) for pattern AD → FD, Cluster 2

User	Day Gap	Comment	Theme
user1	0	@jburns It's intentionally in 4.02 and not schedule for a milestone. I'd like to potentially take another crack at it again in 4.02 at the end in the RC if resources are available or if we institute the run team concept.	AD
user2	268	@user3: is there any idea yet of when this may be implemented?	
user3	0	Probably in the June 2014 release.	
user3	0	FYI to @pwvogel @user4 and @jpwhit that we are getting more inquiries as to when we make this shift. We should look at a exploration in 4.0.6 and look to make the switch in the June 2014 release. At that time we will be looking to bundle WAS Liberty and later versions of IES on the client and server.	
user4	1	Agreed. That's what I've been telling folks (I get inquiries too) - June 2014	
user1	13	@user3 @user4 The System Requirements link https://jazz.net/SystemRequirements says WAS 8.5.5 will be supported as of 4.0.5 which runs on Java 7 , is this information correct?	FD

Table 11 – Continued from previous page

user5	13	"@user4 and @duongn, For RTC Install, we would like to start packaging the Java 7 JDK with the RTC Eclipse 4.2.x client. The main reason for doing this is to unblock the creation of a Mac-based IM install for the RTC client (there is no Java 6 IBM JDK for Mac, but there is a Java 7 for Mac). More details are in these items: - Provide IM based Mac support for RTC client (250364) - see also item 232063, comment 13. This will not affect the server or other clients (like the Eclipse 3.6 client). It also means that RTC would be shipping both Java 6 and Java 7 and the RTC client license would need to reflect that. We would target the same Java 7 build that RAD and RSA 9.x are currently using."
user1	1	@user5 in comment 6 it reads like this is with respect to packaging for 4.0.5. Is that true? It is a late for adding such a change I would think.
user5	0	"@jdgraham, I don't think we have any commitments to add Mac support for IM in 4.0.5, but we do have interest in it. I'm mainly trying to make forward progress on this so that if we miss 4.0.5, we will be in position to finish it in 4.0.6. From chatting with @duongn yesterday, he indicated the changes to the RTC legal text could probably be ready for 4.0.5 RC1 (but not for 4.0.5 Sprint 2). However there could be other aspects of this (like Java cert?) that can't be contained to 4.0.5 at this point."
user4	1	@jdgraham this is NOT for CLM 4.0.5 Note Planned for above (backlog) It would be way too late for 4.0.5 at this point (agreed). I believe the current plan is for Q2 2014. @sandyg - in Clearinghouse it indicates that WAS 8.5.5 supports Java 6 and above. Does NOT require Java 7.
user6	0	@user4 , thanks for clearing that up.... appreciate it.

4.1.3. Cluster 3: Crisis management due to poor understanding of requirements

Main pattern: AD → TD

Reason for delay: Inadequately elicited requirements.

Work items in this cluster are characterized by discussions that indicate mitigation strategies to avoid implementation delays, followed by hasty implementation. The work items are of high priority and need to be addressed in order to unblock a dependent WI or to meet a release candidate. For example, in WI 182270 :

> *@userX Based on feedback from the JAF PMC, this item will take precedence for us in M6, pushing incomplete arch debt work out to M7. This is a very high priority item on the JAF 2012 plan and it is now ready for adoption by the components, so we need to take care of it as soon as possible[...].*

Consequently, a lot of children tasks are created to speed up the implementation of the features.

However, due to poor understanding of the requested functionality, the team is left with little choice and realizes the scope of work is too broad and the development of the specifications takes too much time:

> *@userX - the work on this one is actually in-progress and hopefully will be delivered soon (see the linked child story). We don't really have a choice here since the IBM JRE team is no longer accepting dependencies on Java 6 due to it being EOL by Oracle. I've updated this plan item to indicate the dev commitment level.*
>
> *This is going over the estimate because the migration piece is more complicated than I was hoping it would be.*

The result is providing the minimum necessary implementation to pass the release and push the remaining work into a newly created WI, for a future iteration. The pattern also indicates that there is poor elicitation for these WI in terms of the required features and its importance within the planned release. An example discussion for a WI in this cluster is shown in Table 12.

Table 12: Example WI discussion (WI 187151) for pattern AD → TD, Cluster 3

User	Day Gap	Comment	Theme
user1	0	@user2 I put this in M7. It's probably a stretch for us to get this completely implemented in M7 but hopefully we can do enough that it helps us eliminate any grayness with it even if we hold the delivery back.	AD
user2	35	This is going over the estimate because the migration piece is more complicated than I was hoping it would be.	
user3	14	"This caused 3 new warnings in our tests: DescriptionResourcePathLocationType The method enableTeamAreaReadAccessContext from the type IProcessServerService is deprecatedProcessServiceTests.java. The method isTeamAreaReadAccessContextEnabled from the type IProcessServerService is deprecatedProcessServiceTests.java The method isTeamAreaReadAccessContextEnabled(ITeamAreaHandle, String) from the type IProcessServerService is deprecatedProcessServiceTests.java If the methods are deprecated, seems like the tests should not use them"	TD
user3	0	Also, the @deprecated tag should include information on how to move off of the methods, so our clients can adopt easily.	
user2	3	In addition to these comments, I discovered a more major problem with my changes. I did not update the remove method to account for the new context type and the fact that a group might contain the same item twice with different context types. I will deliver this for M8 if I can implement it tonight.	
user2	0	Actually we do not support the addition of a process area to the same access group twice, which should be fine going forward, so I don't need to worry about the remove case. But I still needed to update the remove method to account for the new context type. And I discovered a junit gap, which I filled in. In the process I saw that we needed to be passing another service (the item service) into ProcessAreaContextUpdater when we instantiate it in AccessGroupService#getContextMembersToAdd.	

4.1.4. Cluster 4: Functionality clarification and rescoping that does no longer fit in current iteration

Pattern 1: Feature clarification (FD)

Reason for delay: Extended functional clarifications without timely decisions.

Table 13 shows an example discussion of this pattern. WI discussions presenting this pattern are representative of solution proposals, clarifications and considerations of tradeoffs in possible design solutions. They usually exhibit some uncertainty regarding the general direction of the work items and corresponding work; the discussions are not about the technical issues, but clarifications on the desired functionality. Agreements on necessary work are reflected in assignments to new children WI and not discussed in these work items.

Draft understanding question of this work item: [...]
What should we do about this? Do we want to [...]? If so there are a couple of issues.
The simplest way to do this is via [...]. I've submitted this as work item 124549.

Table 13: Example WI discussion (WI 166350) for patter FD, Cluster 4

textbfUser	**Day Gap**	**Comment**	**Theme**
user1	0	@user2 @user5 @user3 @user4 Any other ideas?	
user2	0	Is it possible to ship preloaded process descriptions/ practices(exported from RMC) together with PLM and tell the user that they can benefit from PLM?	
user3	0	I think we could provide an article in Jazz.net that describing some best practices how to use process authoring. For the existing templates, their web sites are generated by RMC from libraries, we might provide some exported process description zips for each templates that let user import it directly.	FD
user4	0	My suggestion is provide an extra link in start page to create a sample process description, just like process template(predefined template) and CLM(financial banking application) did. Now we have 3 links Create/Import/Associate , we can add another link named Create a sample process description .	
user4	0	Continued to comment 4, we can do that in PLM as well. We can ship process authoring with some predefined process descriptions (exported from RMC) and import these predefined process descriptions to create samples.	
user2	0	deploy a process description to deploy preload process description? but I don't think we can do that for this release.	
user5	30	If someone already started working on any topic, please comment in this work item so we will not step on each other.	
user4	0	@user5 @user2 @user3 I created a wiki page https://jazz.net/*/ProcessAuthoringUserGuide. I think we can use this wiki page to write the user guides for how to use process authoring in CLM 2011. I have already taken over permission part, viewlet part and started working on these two parts.	
user2	3	I created a wiki page for RMC integration	
user1	5	@user5 Phong, assigning to you to track this. Currently, we have three articles that are written or in progress (the three child tasks of this story). Other ideas for us to consider are blog posting(s), sample process descriptions for customers to import, and maybe a video introduction.	
user5	0	We can make the Eclipse Way process description available for customers to import.	

Pattern 2: Feature rescoping (NM → FD)
Reason for delay: Major modification requirement.

Table 14 shows a typical discussion of this pattern. When this pattern occurs, it represents a required modification to the base functionality of item, not just a technical issue:

More changes required: remove the old unused action label string, add the accelerator for the new menu label [...].

This could be due to a miss understanding of specifications, either by the developer or reviewer:

Maybe I misunderstood; I thought this was for [...].

Table 14: Example WI discussion (WI 52997) for pattern NM → FD

User	Day Gap	Comment	Theme
user1	0	@user2 Please review for delivery.	NM
user2	0	With the move to the New menu, the action string will need to be changed from Create Process Template... to Process Template... to fit with the New menu conventions.	
user3	1	@user2 Implemented the review comments. Ready for review again.	
user2	0	"More changes required: remove the old unused action label string add the accelerator for the new menu label We always strive to have unique accelerators for a menu entry. We also work to keep only the strings that are currently in use as each string needs to be paid to be translated. Added a changeset with these additional changes."	
user2	0	Approved with additional changeset. Please release my changeset as well Shivank. Thanks.	
user3	0	Resolved.	
user4	12	"The new location and label of this action loses any of the connotation that this action will create a process template *from* the project area. Now it just looks somehow misplaced. Normal New actions create a basic item, possibly with some values filled in from the context (e.g. new team area shows the selected project area as the parent area). But this action doesn't create a new process template that has some field initialized to the selected project area: it's almost a kind of export. I don't think the previous wording was great by any means and I'd be happy to see it improved. But this change is a move in the wrong direction."	FD
user3	2	"@user4, @user1: I agree with the comments above. We are not creating a 'new' process template from scratch but duplicating / extracting process configurations of selected project area and wrapping it into a new process template. New action does not give this picture. We should move it back to main context menu and assign it a better caption. I was thinking on the lines of Duplicate Process... Extract Process... Extract Process Configurations... I avoided using keyword 'Process Template' above because it might give a picture that we are trying to duplicate the original process template which was used while creating the selected project area. suggestions? :-) ps: I hope, I am getting it right now after learning about the process world a little better :-)"	
user4	0	I like Extract Process Template... actually.	

4.1.5. Cluster 5: Feature design followed by implementation that extends beyond current iteration

Main Pattern: FD → AD

Reason for delay: Implementation requires additional work in children WIs that are being planned for future iterations.

Work-items in this pattern are typically high level descriptions of functionality, i.e. plan items or stories:

> *@userX,@UserY, @userZ, the issue of whether or not we want to do a bulk role editor came up [...] I think that there are some limitations to doing a bulk editor*

in an iframe that may make us want to reconsider our solution. Implementing a bulk editor natively in LPA might be a better option. What are your thoughts? .

Table 15 illustrates a typical WI in this pattern. Discussion starts on possible ways to implement the feature in terms of feasibility and time constraints. The creation and planning of children tasks is discussed to tackle the agreed part of the feature.

There is however a long gap between this planning discussion and follow up discussions (during this time children WIs are being completed). Yet the planning of the next steps indicates that work will go beyond the current iteration.

Table 15: Example WI discussion (WI 193729) for pattern FD → AD, Cluster 5

User	Day Gap	Comment	Theme
user1	0	"@user2 Is it safe to assume that hasLocalRepository implies local friends storage ?In other words: Applications that are built on the JAF SDK and run in delegated authentication mode must store friend relationships in their local repository"	FD
user2	0	There was a recent email thread on that topic, and I don't think that it was definitely agreed that fronting app friend storage should be local. It pretty much has to be in 4.0, since lots of JAF services expect to fetch friends from the local repository, and there's no time to change that now. So I think the answer should be yes (at least for 4.0). But I'd like to check that @user3 and @user5 agree.	
user3	0	@user2 sorry, the mail thread got buried. Reviewing it now in the context of prepping for DM migration call.	
user3	0	@user6 I agree with comment 2, esp. the at least for 4.0 part, and migration of existing apps that currently have a private JTS. I'm not yet convinced it's what we would want in the future or for new build apps. I doubt we could avoid it for a 4.0-based new build app though without a lot of new API.	
user2	2	In the last DM migration meeting (on 2/1), there seemed to be no objections to the strategy of storing fronting app friends in the fronting app's own repository, so @user1 I believe we can say the answer to your question in comment 1 is yes .	
user4	0	"@user1 - is this task anything more than modifying DelegatedAuthProvider to provide an access to the local friends list, and fixing the jtsConsumer references? I'm noticing the 2w estimate. I'd like to do at least part of this so that I can continue with OAuth 1.0a testing."	

user1	3	@user4 The first change would probably be in AuthenticatingClientService, line 96:
		if (!EnvironmentUtil.isDelegatingAuthentication()) {...}
		This would change back to:
		if (EnvironmentUtil.hasLocalRepository()) {...}	

user2	22	@user1 are you planning on implementing this for 4.0? I believe it will be required for DM.	AD
user5	7	Can someone manually add friends in JTS to workaround this ? Not pretty, but not a blocker then.	
user1	2	We don't really know what is not working at this point. The proxy works in this environment, but there are still other things that are sensitive like theming, dashboards, viewlets, open social support, etc.	

4.1.6. Cluster 6: Technical clarification and coordination leaving no time for finishing the work item

Pattern 1: TD → AD

Reason for delay: Additional planning and coordination required beyond the current iteration. An example discussion is available in Table 16.

Discussions following this pattern show a technical clarification step, followed by coordination between team members for implementation. In some cases, the solution is decided upon or even done but cooperation is required to decide the following steps, like integration or testing.

> *We talked about how to do this today, do you want to push this bug down to repo, or should I open our own to implement it?*
>
> *Had a conversation with @UserX, he will get back to us on this after testing in his dev environment.*

Table 16: Example WI discussion (WI 67521) for pattern TD → AD, Cluster 6

User	Day Gap	Comment	Theme
user1	0	Can you elaborate on why this is different from any of the other caches?	TD
user2	0	"Its not different, it just requires some different update paths in order to maintain transaction safety. You cannot use volatile fields for concurrency reasons if updating the value of the field needs to know the previous value of the field. To update a list (add/remove), you need to know the previous value of the list."	
user3	0	"You can use volatile fields, that isn't the problem. The problem is you want to update a collection in a transaction safe way. The way it currently is with transactional cachei f you start with t1 adds a meanwhile t2 adds b. t1 commits so now the collection has a if somebody fetches the collection at this point it will be marked as the current value. If t2 now commits, it will commit b, and now the cache is wrong."	
user1	0	How would that work with items? It would throw a stale data exception when t2 attempts to commit. Why should this be different?	
user3	0	"Because you want concurrent access to the collection. It should be fine for 2 transactions to add values to it simultaneously."	
user1	0	Ok, I just took a look with Balaji, because he has similar requirements with the cache to be used by the context manager service (user->list of context ids). We should collaborate on the solution so we don't overlap. If you guys propose something before we do, just let us know.	AD
user4	29	We talked about how to do this today, do you want to push this bug down to repo, or should I open our own to implement it?	

Pattern 2: Implementation hesitation(TD)

Reason for delay: Extended technical clarifications.

An example discussion for this pattern is shown in Table 17. Items falling under this pattern show a lack of confidence in how to approach the implementation of the solution. Usually the discussions are centered around different proposals towards the same goal, weighing the implications of each, or speculation regarding the ambiguous requirement or specification.

> *The one question I have is whether we want to include the new zOS steps there.*

I am thinking something in between Option 3 or 4 could be envisaged. If so, then Option 4 could consist in [...]. This might however not perform well. Option 3 is more work in [...], but we should explore that path.

Table 17: Example WI discussion (WI 90783) for pattern TD, Cluster 6

User	Day Gap	Comment	Theme
user1	0	This would come from your working directory argument in your server launch config.	TD
user2	2	@user3, JFS needs to create these directories for query and search. There are config props for this - but the default value will be the under the working directory.	
user3	0	"We do not access JFS at all in our tests, I am not sure why this is being eagerly activated. It gets tiring updating all the launch configs every time something like this pops up. (I have over 100, for different databases, test suites, configurations). Why can't you use /tmp in the test context?"	
user3	0	Or alternatively, we add things to jazzignore for all plugins which contain launches.	
user2	2	re: using /tmp in the test context - how do we detect that? Yet another property?	
user3	0	If the property is unspecified, you could put it in a subdirectory of /tmp and print a warning as we do for the versioned content service.	

Pattern 3: Additional effort before integration/delivery (TD → RD and TD → NM)
Reason for delay: No time left for implementation review and modifications.

Conversations following these two patterns are relatively similar in that they require work just before the assigned deadline. The solution itself is implemented, on time, but it either needs to be approved through a code review or it requires modifications following a code review for the delivery to take place:

Could you review the changes I made for [...] ?

@User, for the text of the link to configure for time periods, we decided to go with the general: [Item1], [Item2].

Table 18 shows a typical discussion of this pattern.

Table 18: Example WI discussion (WI 169891) for pattern TD → RD, Cluster 6

User	Day Gap	Comment	Theme
user1	0	"JFS provides such a command, and a mean to repair the orphan data. Why is this needed for 3.0.1.1 ?"	TD
user2	0	@user1, what command are you talking about? The only related commands I see in my workspace are ListStorageAreaKeysCommand and SetStorageAreaKeysCommand, which deal with oAuth keys and not application keys. Right now we need to get this into 3.0.1.1 because it is possible that any customer using a 3.0 or 3.0.1 server can lose their data as discussed in the parent defect 168625 comment 16 & 18.	

Table 18 – Continued from previous page

user2	12	"We'll need 5 strings translated for the new repotools command. Two of these strings already exist (the log and teamserver.properties parameter descriptions), but it seems the standard is to create a new property in the plugin.properties file for the specific command. Command Strings: Command description Parameter description - log file path (5 words) Parameter description - teamserver.properties (5 words) Output Strings: Invalid Application Ids: Associated Project Areas: [x] more..."	
user2	1	@user3, I've attached a completed changeset with the necessary strings for the command, because strings needing translation are due by tomorrow night's 3.0.1.1 build. Could you take a look and make sure the Messages files are structured properly, and that the messages themselves make sense?	RD
user2	25	@user4 The command implementation is basically the same as when we went over it last week; The only change I put in was to add a new error message and log statement to the query service catch statement in queryForInvalidIds.	

4.2. Comparative Analysis of Late and Non-late Work Items

Table 19 outlines the results of the cluster analysis of the time-series of the *non-late* work items. We conducted a similar analysis of the 1) semantics in the WI discussions within each cluster to discover patterns across the respective WI time series, 2) time stamps of comments and 3) number of child tasks created during the WI iteration as we did for the late work items.

Across all non-late WI we identified one predominant pattern in the evolution of their discussion over time, and one that is much different than those in the late WIs. We describe the pattern below, though there are certain interesting differences between the late and non-late WI discussions along the other two dimensions that merit highlighting first:

1. *The non-late WIs have many more comments and much higher frequency of communication.* The 125 non-late WI have a total of 4068 comments vs 2655 in the late WIs. Figure 4 and Figure 5 show the WI distribution based on the average number of days between their comments, and number of comments respectively for these 125 late and 125 non-late work items sets. The average number of days between comments, for a WI, is obtained by dividing the number of elapsed days from the first to last comment by the number of comments. The histogram indicates that the late WIs discussions are more spread in time whereas the non-late WIs discussions are more dense in shorter intervals.

2. *Discussions on non-late work items result in much more work delegated to children tasks for implementation.* The 125 late WIs have associated with them a total of 253 child tasks, whereas the non-late ones have a total of 437. A paired-samples t-test of significance in the number of children for late WIs ($M = 2.02; SD = 3.68$) vs. non-late WIs ($M = 3.50; S = 3.94$) yields a significant difference at $p = 0.005$, suggesting that the number of children does have an effect on WI timely completion.

Finally, our analysis of the sequence of codes and the semantics of the WI discussions in these clusters yielded a single predominant pattern across clus-

ters, one that was much different than those in the late WIs. Across all these non-late WIs there is *a cyclical sequence of codes* aligned with agile development and indicating alternating discussion topics of feature clarification, feature design, technical discussions and clarifications and so on.

The characteristics of these discussions include:

1. A request is typically answered quickly, with several comments in the span of a few hours indicating an active conversation followed by an implementation period with reduced comment activity.

2. Misunderstandings, ambiguous requirements or development inactivity are typically resolved by the intervention of a senior developer or manager, offering the needed guidance. This intervention does not necessarily provide a clear solution, but is often a guided discussion towards a resolution or a consultation with individuals outside of the platform, e.g.:

@User Can you review this and get it into process component once we have a successful build/tests.

I brought this WI up at today's PLE Design UI review. The consensus was: [...]

3. The implementation scope is assessed early and narrowed down if necessary, as it can be seen by the multiple occurrence of TDs (Technical discussion) and FDs (Functionality discussion) sequence of codes in table 19.

4. Reports of decisions made external to the communication platform are included to avoid delays in the WI implementation, e.g.:

Talked to @User in person, so removing his approval (approved verbally).

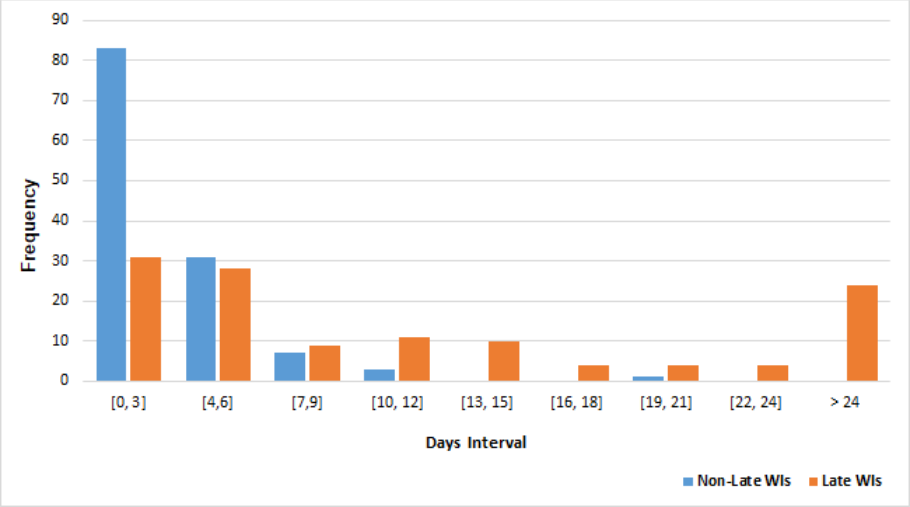

Figure 4: Work items distribution based on the average number of days between comments. *Average number of days between comments*, for a work item, is obtained by dividing the number of elapsed days between first and last comment by the number of comments

Table 19: Clusters of non-late work items. Example members choose randomly and their respective number of comments and time series. The bold item is the cluster centroid.

Cluster	**WI#**	**#Com.**	**Time series**
Cluster 1	**395581**	**25**	**AD FD TD TD**
35 time series	302249	15	AD FD TD AD TD
	393762	19	FD AD FD TD AD
	391076	40	FD AD TD TD FD NM
	392077	27	AD TD AD FD AD NM FD TD
	831210	44	AD NM FD TD AD FD AD NM FD TD AD
Cluster 2	**200493**	**42**	**AD FD AD FD AD**
27 time series	303580	26	AD FD AD FD AD FD
	377366	31	AD FD FD AD NM NM AD TD FD
	243986	23	FD AD FD AD TD FD AD TD AD
	388777	33	AD FD AD FD AD FD AD TD AD FD
	337404	37	AD FD AD TD AD TD AD TD AD FD AD FD
Cluster 3	**244242**	**39**	**FD TD FD TD FD AD**
30 time series	310689	23	FD FD TD FD AD FD AD AD
	244209	58	TD FD TD AD FD TD FD TD AD
	169933	43	AD FD TD FD OP TD FD NM TD AD
	245250	60	FD TD NM FD TD AD FD TD FD TD AD
	342815	52	FD TD AD TD FD TD AD FD TD FD TD AD
Cluster 4	**310617**	**47**	**FD AD TD AD FD**
21 time series	342604	38	FD TD AD FD AD TD
	169933	43	FD AD TD AD FD TD AD
	356801	35	FD AD TD AD FD AD NM TD
	266362	69	TD AD OP TD AD FD AD FD TD
	341514	52	AD FD AD NM TD AD FD AD FD TD
Cluster 5	**399037**	**19**	**TD AD TD**
12 time series	380891	15	TD AD TD
	320721	16	FD OP AD TD
	395618	17	TD AD TD AD
	305281	25	TD AD TD AD TD
	289587	39	TD AD TD AD FD OP AD TD

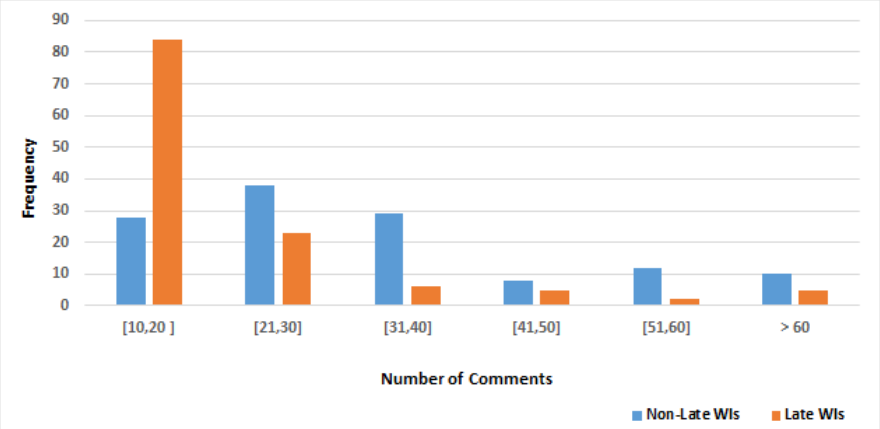

Figure 5: Late and non-late Work items distribution based on their number of comments

5. Discussion

5.1. Results

Our analysis of online developer discussions and clusters of similar WI discussions revealed six patterns of WI delay. The must surprising finding was the strong consistency between the structural (WI time series representation) and the semantic (WI discussion content) patterns in the late WI clusters. WI discussions with high structural similarity (WI in the same cluster) shared the same reason for delay. Although having an homogeneity in a cluster is the goal of clustering, high semantic consistency within WI clusters was unexpected. This behavior suggest consistent interaction dynamics among the developers. The major contrasting points in our analysis between the late and non-late work items are the communication and the tasks dispatch management.

5.1.1. Communication is Key!

Communication in a project is important and even more when a group has to evolve towards a common objective. Our analysis reveals that the coordination and planning around WI that are late take too long to result in agreement. Discussions are slower and the responses to requests are not handled fast enough, leading to finishing the implementation beyond the planned iteration. Another communication issue noticed is the endless planning and rescoping discussions around the late WIs. Cluster1 in the late WIs is a typical example. Some patterns in these late WI discussions indicate that they appear to suffer from decision paralysis. Often stakeholders continue to clarify the requirement because it is ambiguous, incomplete, or has frequent changes. As a result, its implementation can be delayed or sometimes never get started. Bikeshedding, also known as the Parkinson's law of triviality [48, 42] is another common situation in which developers give disproportionate weight and time to solving trivial issues and delay development.

In contrast, our analysis of comment time-stamps shows that the non-late WI are characterized by communication that is quite frequent and with requests being processed quickly and effectively. This suggests that the fast responses and feedback present in conversations are a possible reason for the timely resolution of these WI. Discussions are fluid and requests are addressed promptly to avoid blocking the progression of the WI implementation. As a result, developers and team leads are able to adjust the WI scope, and easily divide the task to the children without coordination overhead. This result suggests that the design of future collaboration tools can include digital nudges to help developers become aware when to increase the frequency of communication about particular WIs and therefore possibly reducing the delay of feedback in the project. Digital nudging describes "the use of user-interface design elements to guide people's behavior in digital choice environments" [64]. As such, we believe, labeling/nudging (automatically or manually) messages about the *severity* (blocker or not) and the *priority* (emergency or not) of WIs will increase the possibility of handling blocking WIs and thus reducing the overall delay of the project.

5.1.2. Task Management Matters!

Our analysis suggests that the way and frequency with which WI tasks were managed was different in the late vs. not late WIs. In non-late WIs the elicitation of requirements appears to have been more thorough, resulting in an easier process of dividing tasks to children. Moreover, given the frequency of responses in the WI discussions, actively rescoping the functionality and eventually delegating some to the children tasks in order to meet the objectives was possible. In contrast, the discussions in the late WIs suggest the intention to fulfill all the requirements at once in the implementation of one single WI without much delegation to children tasks. This behavior is likely the result of too much time spent in clarifying and eliciting further information on the WI, without much time left for its implementation review or modifications.

5.1.3. What About Technical Debt?

Introduced by Cunningham [13], technical debt explains the need for refactoring, and the impact of design choices on a software product. Research on technical debt has since explored and studied the metaphor to explain [60], assess [23] , manage [41], or understand the impact [50] of technical debt on the organization productivity. Technical debt relates to the additional cost and rework over the software life cycle when a short-term, easy solution is chosen instead of a better solution. As such, it conceptualizes the trade off between the short-term benefit of rapid delivery and long-term value. Understanding and managing technical debt is an important goal for many organizations. Not all debt is bad, and if incurring some technical debt helps your company achieve a big goal, then it could be worth the "interest payment" of a more difficult task of future software updates or adding new features. But problems do arise, and what may seem like minor annoyances now often become major issues if left alone for too long.

In the IBM project we studied, we notably identified two kinds of technical debt:

- *Planned technical debt*, when the team is challenged by the iteration deadline and forced to choose to implement high priority features and to push the remaining features into the next iterations by creating new work items to track the debt. For instance, *Cluster 3* for *Crisis management* is a perfect illustration. This kind of technical debt, despite the additional work carried over to the upcoming iterations, has the advantage of not being "forgotten" to be paid, i.e., the features are saved for future implementation.

- *Unplanned technical debt*, when a workaround (quick and easy solution) is provided in order to deliver a blocking or high priority feature. Unlike the previous one, this kind of debt is not easy to track, as there is no creation of WI to "remember" the debt and the team moves quickly/forward to the implementation of new features. The major consequence is that the debt is forgotten and grows up to becoming a liability for the system maintainability and scalability.

Our analysis shows that both of these two types of debt are found in both late and non-late WIs, even though they are more recurrent in late ones. A worthwhile future research direction is to investigate whether late WIs are in fact *consequences-of/related-to* techical debt in non-late WIs (i.e. the cause of/correlated to), and whether they would be planned or unplanned technical debt.

5.2. Implications

5.2.1. For Research

To the best of our knowledge, our work is one of the first to use time-series and clustering techniques on results of thematic analysis on developer conversations. We believe there are a number of new areas worthy of further exploration, guided by questions that include:

- Is it possible and how to automatically segment conversation threads with respect of the theme discussed? Themes identification in thematic analysis is very time consuming. Currently it needs to be carried out manually since it can cover multiple comments and there is more than one theme in one conversation.

- Could sentiment analysis technique be employed to enhance the analysis outcomes?

- What features could complement text data to not only characterize the development but also predict the impact on the iteration and its deadline?

All those questions highlight natural language processing and machine learning challenges for design and evaluation of automated means to identify the patterns.

5.2.2. For Practitioners

Our work has implications for tools that automatically support iteration planning and monitor development progress for project managers. Such tools can actively analyze the conversations developers carry on particular WIs and provide profiles of WIs and their progress. The profile can suggest whether the WI implementation process has issues through the identification of codes (AD, FD, ...) and their sequences, and recommend the potential impact on meeting the planned implementation deadline. It will help managers to be proactive by quickly reacting on blocking items and constantly tracking the milestones and adjusting accordingly to the deadline. Since the automated analysis involves textual data, the more data is formatted (e.g. tags, sentences without slang) the better the automation process can be set up.

5.3. Threats to validity

5.3.1. Misuse of tracking platform

One controversial point of using repository data is the extent to which the platform is correctly used or, misused. It is easy to imagine submitting statistics or tracking information in such a way as to artificially improve certain performance indicators or obscure latent problems. This type of practice represents a difficult to overcome hurdle as we have no other way of analyzing a past situation but through data that is, potentially, misrepresenting the actual reality of the situation and its context.

5.3.2. Communication outside the platform

As the tracking platform is only one of the tools used for communication within the project team, some information is impossible to capture. Teams share information in an informal, face-to-face manner or by means of calls or emails that are not present in our data-set. We have encountered comments such as 'After the discussion with [...]' clearly indicating that decisions were outside of the platform. Even though the result is sometimes communicated and logged, we cannot ensure how many other undisclosed conversations have an impact on the project, an impact that is hard to track and analyze.

5.3.3. Too high abstraction level

The first step in our process consisted of labeling each comment in WI discussions with a theme best describing its content. When a comment was related to more than one theme, we assigned it, for the sake of consistency in our methodology, to the predominant theme. This level of abstraction may have removed critical information necessary for accurate analysis and interpretation. Moreover during the semantic analysis, it was observed that structural sequences that are very similar can have opposing semantic interpretation, for example a work status report possibly relating to success or a failure in one's endeavor.

6. Conclusion

Our analysis has showed that text data generated along side software development process is a mine of useful information about the progression in a requirement implementation and the dynamics of developers interaction to address work items efficiently. We used time series to model the chronological evolution of work items' comments chain and with the help of clustering algorithm to group them by structural similarity. The semantic analysis of clusters give an insightful explanation of the delay reasons. We found that late WI exhibit different archetypes of delay and each is associated with a specific reason why the delivery of the requirement is getting late. The common reason for the delay is a lack of fluent communication associated with a poor project management. Conversely non-late WI delegate more to children tasks and are proactive on handling requests.

These finding has a potential to be used to monitor workflow, resolve the knot points quickly, and for more active team management. The comment chains annotations step was done manually. We think there are machine learning and natural language processing challenges here to tackle in our future projects. We also believe that providing communications happening outside the tracking platform (calls, meeting transcripts) can increase the accuracy of the data view.

References

[1] Sousuke Amasaki, Takashi Yoshitomi, Osamu Mizuno, Yasunari Takagi, and Tohru Kikuno. A new challenge for applying time series metrics data to software quality estimation. *Software Quality Journal*, 13(2):177–193, 2005.

[2] Ayman Amin, Lars Grunske, and Alan Colman. An approach to software reliability prediction based on time series modeling. *Journal of Systems and Software*, 86(7):1923–1932, 2013.

[3] Preeti Arora, Shipra Varshney, et al. Analysis of k-means and k-medoids algorithm for big data. *Procedia Computer Science*, 78:507–512, 2016.

[4] Ziv Bar-Joseph. Analyzing time series gene expression data. *Bioinformatics*, 20(16):2493–2503, 2004.

[5] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. The agile manifesto, 2001.

[6] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[7] Serdar Biçer, Ayşe Başar Bener, and Bora Çağlayan. Defect prediction using social network analysis on issue repositories. In *Proceedings of the 2011 International Conference on Software and Systems Process*, pages 63–71. ACM, 2011.

[8] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006.

[9] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, and Aditya Ghose. Predicting the delay of issues with due dates in software projects. *Empirical Software Engineering*, 22(3):1223–1263, 2017.

[10] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Aditya Ghose, and John Grundy. Predicting delivery capability in iterative software development. *IEEE Transactions on Software Engineering*, 2017.

[11] Tak chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164 – 181, 2011.

[12] D. S. Cruzes and T. Dybâ. Recommended steps for thematic synthesis in software engineering. In *2011 International Symposium on Empirical Software Engineering and Measurement(ESEM)*, volume 00, pages 275–284, 09 2011.

[13] Ward Cunningham. The wycash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2):29–30, 1992.

[14] Subhajit Datta, Renuka Sindhgatta, and Bikram Sengupta. Talk versus work: characteristics of developer collaboration on the jazz platform. In *ACM SIGPLAN Notices*, volume 47, pages 655–668. ACM, 2012.

[15] Ali Dehghan, Adam Neal, Kelly Blincoe, Johan Linaker, and Daniela Damian. Predicting likelihood of requirement implementation within the planned iteration: an empirical study at ibm. In *Mining Software Repositories (MSR), 2017 IEEE/ACM 14th International Conference on*, pages 124–134. IEEE, 2017.

[16] Lydia DeSantis and Doris Noel Ugarriza. The concept of theme as used in qualitative nursing research. *Western Journal of Nursing Research*, 22(3):351–372, 2000. PMID: 10804897.

[17] SM Didar Al Alam, Muhammad Rezaul Karim, Dietmar Pfahl, and Günther Ruhe. Comparative analysis of predictive techniques for release readiness classification. In *Proceedings of the 5th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, pages 15–21. ACM, 2016.

[18] Jason Ernst, Gerard J Nau, and Ziv Bar-Joseph. Clustering short time series gene expression data. *Bioinformatics*, 21(suppl_1):i159–i168, 2005.

[19] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45:12, 2012.

[20] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. *Fast subsequence matching in time-series databases*, volume 23. ACM, 1994.

[21] Jennifer Fereday and Eimear Muir-Cochrane. Demonstrating rigor using thematic analysis: A hybrid approach of inductive and deductive coding and theme development. *International Journal of Qualitative Methods*, 5(1):80–92, 2006.

[22] R. Frost. Jazz and the eclipse way of collaboration. *IEEE Software*, 24(6), Nov 2007.

[23] I Gat. Technical debt assessment: A case of simultaneous improvement at three levels. *Agile Product & Project Management Advisory Service Executive Update*, 11, 2010.

[24] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of intelligent information systems*, 17:107–145, 2001.

[25] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[26] Iain Hay. Qualitative research methods in human geography. *Oxford University Press*, 2000.

[27] V. T. Heikkilä, D. Damian, C. Lassenius, and M. Paasivaara. A mapping study on requirements engineering in agile software development. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, pages 199–207, Aug 2015.

[28] Hsiu-Fang Hsieh and Sarah E Shannon. Three approaches to qualitative content analysis. *Qualitative health research*, 15(9):1277–1288, 2005.

[29] Yan-Ping Huang, Chung-Chian Hsu, and Sheng-Hsuan Wang. Pattern recognition in time series database: A case study on financial database. *Expert Systems with Applications*, 33(1):199–205, 2007.

[30] Hesam Izakian, Witold Pedrycz, and Iqbal Jamal. Fuzzy clustering of time series data using dynamic time warping distance. *Engineering Applications of Artificial Intelligence*, 39:235–244, 2015.

[31] Teemu Karvonen, Woubshet Behutiye, Markku Oivo, and Pasi Kuvaja. Systematic literature review on the impacts of agile release engineering practices. *Information and Software Technology*, 86:87 – 100, 2017.

[32] David Kavaler, Sasha Sirovica, Vincent Hellendoorn, Raul Aranovich, and Vladimir Filkov. Perceived language complexity in github issue discussions and their effect on issue resolution. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 72–83. IEEE Press, 2017.

[33] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. Segmenting time series: A survey and novel approach. In *Data mining in time series databases*, pages 1–21. World Scientific, 2004.

[34] Hyun Duk Kim, Malu Castellanos, Meichun Hsu, ChengXiang Zhai, Thomas Rietz, and Daniel Diermeier. Mining causal topics in text data: Iterative topic modeling with time series feedback. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 885–890, New York, NY, USA, 2013. ACM.

[35] Eric Knauss, Daniela Damian, Jane Cleland-Huang, and Remko Helms. Patterns of continuous requirements clarification. *Requirements Engineering*, 20(4):383–403, 2015.

[36] Helena C Kraemer. Kappa coefficient. *Wiley StatsRef: Statistics Reference Online*, pages 1–4, 2014.

[37] Dan Li, Kerry D Wong, Yu-Hen Hu, and Akbar M Sayeed. Detection, classification, and tracking of targets. *IEEE SIGNAL PROCESSING MAGAZINE*, 2002.

[38] Sherlock A Licorish and Stephen G MacDonell. Exploring the links between software development task type, team attitudes and task completion performance: Insights from the jazz repository. *Information and Software Technology*, 2017.

[39] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[40] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*, 2017.

[41] Steve McConnell. Managing technical debt. *Construx Software Builders, Inc*, pages 1–14, 2008.

[42] Paul Mcfedries. Agile words [technically speaking]. *IEEE Spectrum*, 54(6):21–21, 2017.

[43] Monnie McGee and Ian Harris. Coping with nonstationarity in categorical time series. *Journal of Probability and Statistics*, 2012, 2012.

[44] Andrew Meneely, Laurie Williams, Will Snipes, and Jason Osborne. Predicting failures with developer networks and social network analysis. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pages 13–23. ACM, 2008.

[45] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010.

[46] Vit Niennattrakul and Chotirat Ann Ratanamahatana. On clustering multimedia time series data using k-means and dynamic time warping. In *Multimedia and Ubiquitous Engineering, 2007. MUE'07. International Conference on*, pages 733–738. IEEE, 2007.

[47] John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1855–1870. ACM, 2015.

[48] Cyril Northcote Parkinson and Osbert Lancaster. *Parkinson's Law or the Pursuit of Progress.* Murray London, 1958.

[49] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

[50] Ken Power. Understanding the impact of technical debt on the capacity and velocity of teams and organizations: Viewing team and organization capacity as a portfolio of real options. In *2013 4th International Workshop on Managing Technical Debt (MTD)*, pages 28–31. IEEE, 2013.

[51] Tomas Pranckevičius and Virginijus Marcinkevičius. Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. *Baltic Journal of Modern Computing*, 5(2):221, 2017.

[52] Uzma Raja, Joanne Elaine Hale, and David Peter Hale. Temporal patterns of software evolution defects: A comparative analysis of open source and closed source projects. *Journal of Software Engineering and Applications*, 4(08):497, 2011.

[53] Scott Rich. Ibm's jazz integration architecture: building a tools integration architecture and community inspired by the web. In *Proceedings of the 19th International Conference on World wide web*, pages 1379–1382. ACM, 2010.

[54] Christian Richter, Martin Luboschik, Martin Röhlig, and Heidrun Schumann. Sequencing of categorical time series. In *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 213–214. IEEE, 2015.

[55] Pedro Pereira Rodrigues, João Gama, and Joao Pedroso. Hierarchical clustering of time-series data streams. *IEEE transactions on knowledge and data engineering*, 20(5):615–627, 2008.

[56] H.C. Romesburg. *Cluster analysis for researchers*. Lifetime Learning Publications, 2004.

[57] Johnny Saldaña. *The coding manual for qualitative researchers*. Sage, 2015.

[58] Ken Schwaber and Jeff Sutherland. The scrum guide - the definitive guide to scrum: The rules of the game, November 2017.

[59] Jonathan A Smith. *Qualitative psychology: A practical guide to research methods*. Sage, 2015.

[60] C Sterling. Chapter 2: Technical debt. *Managing Software Debt: Building for Inevitable Change*, 2011.

[61] Mojtaba Vaismoradi, Hannele Turunen, and Terese Bondas. Content analysis and thematic analysis: Implications for conducting a qualitative descriptive study. *Nursing & Health Sciences*, 15(3):398–405, 2013.

[62] Jun Wang, Arvind Balasubramanian, Luis Mojica de la Vega, Jordan R Green, Ashok Samal, and Balakrishnan Prabhakaran. Word recognition from continuous articulatory movement time-series data using symbolic representations. *SLPAT 2013*, 2013.

[63] Andreas S Weigend. *Time series prediction: forecasting the future and understanding the past*. Routledge, 2018.

[64] Markus Weinmann, Christoph Schneider, and Jan vom Brocke. Digital nudging. business & information systems engineering 58, 6 (dec. 2016), 433–436, 2016.

[65] Christian H Weiß. Continuously monitoring categorical processes. *Quality Technology & Quantitative Management*, 9(2):171–188, 2012.

[66] Gary M Weiss. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter*, 6:7–19, 2004.

[67] Timo Wolf, Adrian Schroter, Daniela Damian, and Thanh Nguyen. Predicting build failures using social network analysis on developer communication. In *Proceedings of the 31st International Conference on Software Engineering*, pages 1–11. IEEE Computer Society, 2009.

[68] Nong Ye, Syed Masum Emran, Qiang Chen, and Sean Vilbert. Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Transactions on computers*, 51(7):810–820, 2002.

[69] Murat Yilmaz, Rory V. O'Connor, Ricardo Colomo-Palacios, and Paul Clarke. An examination of personality traits and how they impact on software development teams. *Information and Software Technology*, 86:101 – 122, 2017.

[70] Kyung-A Yoon, Oh-Sung Kwon, and Doo-Hwan Bae. An approach to outlier detection of software measurement data using the k-means clustering method. In *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, pages 443–445. IEEE, 2007.

[71] Shi Zhong, Taghi M Khoshgoftaar, and Naeem Seliya. Analyzing software measurement data with clustering techniques. *IEEE Intelligent Systems*, 19(2):20–27, 2004.