Managing Technical Debt in a Multidisciplinary Data Intensive Software Team: an Observational Case Study

Ulrike M. Graetsch[a,*], Rashina Hoda[a], Hourieh Khalazjadeh[b], Mojtaba Shahin[c], John Grundy[a]

[a]*Faculty of Information Technology, Monash University, Wellington Road, CLAYTON, 3800, Victoria, Australia*
[b]*School of Information Technology, Deakin University, 221 Burwood Highway, BURWOOD, 3125, Victoria, Australia*
[c]*School of Computing Technologies, RMIT University, 124 La Trobe Street, MELBOURNE, 3000, Victoria, Australia*

Abstract

Context: There is an increase in the investment and development of data-intensive (DI) solutions – systems that collect, manipulate and visualise large amounts of data. Without careful management, this growing investment will also grow associated technical debt. Delivery of DI solutions requires a multidisciplinary skill set, but there is limited knowledge about how multidisciplinary teams develop DI systems and manage technical debt.
Objective: Contribute empirical, practice based insights about multidisciplinary DI team technical debt management practices.
Method: Exploratory observation case study. We use socio-technical grounded theory to develop concepts and categories that articulate technical debt and technical debt management practices.
Results: We identify technical debt that the DI team deals with, in particular technical data debt and pipeline debt. We explain how the team manages the technical debt, assesses technical debt based on the context in which it is identified, what treatment types they consider and how they implement technical debt treatments to fit sprint capacity constraints.
Conclusion: We discuss implications of the findings and highlight the need for technical debt treatment implementation patterns as well as tool support for multidisciplinary DI teams.

Keywords: Technical Debt, Data-Intensive systems, Data Visualisation, Data Pipelines, Agile Data Delivery, Multidisciplinary teams, Observation Case Study, Socio-Technical Grounded Theory

1. Introduction

Data intensive (DI) solutions are systems that analyse and manipulate data to provide predictions and insights. These systems are becoming pervasive. The big data and business analytics market was valued at US$215.7B in 2021 (Kenyon, 2021) and is projected to be $924B by 2032 (Fortune, 2023). Delivery of DI solutions involves expertise and skills from multidisciplinary teams (Graetsch et al., 2023). DI systems rely on data, not just algorithms or programs to deliver an outcome or result. Examples of DI systems include enterprise data analytics solutions, imaging diagnostic systems, and real estate price predictors. The development of DI systems requires the combination of different disciplines including substantial domain knowledge, software engineering, data science and cloud computing. Traditionally, teams are characterised as multidisciplinary teams because team members present with different bodies of knowledge, research communities, ways of working, education and career pathways. (Choi and Pak, 2006). Within DI systems delivery, cross-skilling between team members of different disciplines is valued, but difficult to achieve (Graetsch et al., 2023). There is growing recognition of the critical role these teams will play in the success of future digital transformation (Gupta, 2022), yet there is limited empirical research in the software engineering domain with focus on these teams, their team members or their ways of working. Research to build this understanding can be challenging. It involves people – often professional practitioners and dealing with organisations that have goals other than research, and requires adherence to ethics processes. Consequently, it is carried out less frequently than other empirical research (Storey et al., 2020).

In parallel with the explosive DI systems growth, there is a growing awareness of the potential for the associated technical debt to grow. Technical debt, first articulated by Cunningham in 1992 (Cunningham, 1992) is a metaphor for describing the 'debt' or effort that is incurred when shortcuts to quality are taken to achieve delivery goals. Technical debt has been a mainstay in software engineering for some time. However, the extension of technical debt to DI systems has been much slower – with the first references by Weber et al. (Weber et al., 2014) and Anice et al.(Aniche et al., 2014) in 2014. A body of research is slowly building including work to characterise and define DI technical debt (Weber et al., 2014; Foidl et al., 2019; Sculley et al., 2015), measuring DI technical debt (Sklavenitis

*Corresponding author
Email addresses: ulrike.graetsch@monash.edu (Ulrike M. Graetsch), rashina.hoda@monash.edu (Rashina Hoda), hkhalajzadeh@deakin.edu.au (Hourieh Khalazjadeh), mojtaba.shahin@rmit.edu.au (Mojtaba Shahin), john.grundy@monash.edu (John Grundy)

and Kalles, 2024) and managing DI debt (Albarak and Bahsoon, 2016, 2018; Albarak et al., 2020; Muse et al., 2022b,a; Kleinwaks et al., 2023; Waltersdorfer et al., 2020).

This research builds knowledge to contribute and extend knowledge about DI technical debt and its management and builds on our prior research into the challenges that multidisciplinary DI teams face and how they deal with data challenges (Graetsch et al., 2023). We aim to deepen understanding of multidisciplinary DI team data practices through detailed observations, including their practices relating to technical debt. We designed an exploratory observation case study to observe a 12 member agile data analytics delivery team in a large organisation. We observed the team for 6 weeks and collected data for analysis, including rich descriptions of the case context. We applied socio-technical-grounded-theory method (Hoda, 2024) for data analysis to develop concepts and categories about technical debt and its management and present these in our findings. We discuss the findings in the context of related literature. Key findings of the study include details about the technical debt that DI teams discuss, and the team's practices such as assessing technical debt and structuring work to address technical debt during agile scrum ceremonies. The team balances their desire for holistic solutions with the reality of limited capacity, using splitting strategies to achieve treatment of technical debt so they can deliver within the boundaries of their sprint capacity. We highlight the need for new patterns to pay down or treat technical debt that can be adapted to multidisciplinary DI system delivery.

This research makes the following key contributions:

- We present a detailed observational case study that provides descriptions and contextual information about multidisciplinary team member work practices in an agile data-analytics team.
- We conceptualise technical debt related discussions in a multidisciplinary data-analytics team, including the contexts in which technical debt is identified and assessed, how it assessed and treatment approaches considered.
- We provide a set of recommendations for practitioners and key future work directions for researchers.

The rest of this paper is organised as follows: We present related works in Section 2. Section 3 provides an overview of our research method. We present our study findings in Section 4 and discuss our findings in terms of the related works in Section 5 where we also present key implications and recommendations for both practitioners and researchers. Section 6 considers the threats and limitations of our study and Section 7 provides a concluding summary.

2. Background and Motivation

2.1. Technical Debt and Technical Debt Management

Technical debt, first articulated by Cunningham in 1992 (Cunningham, 1992) is a metaphor for describing the 'debt' or effort that is incurred when shortcuts to quality are taken to achieve delivery goals. The resulting growth in empirical research led to systematic reviews and early conceptualisations of technical debt (Li et al., 2015; Tom et al., 2013). The Dagstuhl Seminar 16162 (Avgeriou et al., 2016) developed a research roadmap to develop two view points of describing technical debt for software-intensive systems; firstly to explore technical debt properties, artifacts and elements, and secondly, to explore the management of technical debt encompassing the process related activities or different states the debt may go through (Avgeriou et al., 2016).

Technical debt and technical debt management have been systematically analysed from the perspective of traditional software engineering (Li et al., 2015; Alves et al., 2016; Rios et al., 2018; Tom et al., 2013), with Rios et al. synthesising the prior research through their tertiary study (Rios et al., 2018). They also assessed activities, tools and strategies to support technical debt management activities and categorised these into 4 macro TDM activities: prevention, identification, monitoring, and payment. A case study by Guo et al. (2016) studied the effects of implementing a technical debt management approach to better understand the cost and benefit of explicitly managing technical debt, to contribute to the development of a theory about technical debt management (Seaman and Guo, 2011). More recently, Freire et al. (2023) conducted a survey study of software practitioners gained insights in the technical debt payment methods and reasons for not repaying technical debt. They identified that technical debt management is complex and involves a combination of managerial and technical practices, making it important for managers and technical team members to be involved in technical debt repayment (Freire et al., 2023). These foundational works and more recent works were focused on software intensive systems and did not include technical debt considerations for data-intensive systems.

2.2. Data-intensive Software System Technical Debt

Two of the earliest data related technical debt challenges were about missing referential integrity constraints at the database level (Weber et al., 2014) and placing Data Access Object methods into the wrong class (Aniche et al., 2014). The articulation of specific technical data debt was followed by development of a taxonomy of database design related technical debt (Albarak and Bahsoon, 2016).

Sculley et al. raised the concept of specialised technical debt in machine learning systems, specifically due to their data-intensive nature. They identified debt at the system level, as well as data debt in machine learning systems (Sculley et al., 2015). A conceptual model to articulate where data-intensive technical debt can emerge was articulated by Foidl et al. (Foidl et al., 2019). The model shows that within a Data Intensive Software System (DISS), technical debt emerges due to software architecture or software implementation decisions, as well as Data Model or Data Storage constructs, and/or through data - specifically data quality issues (Foidl et al., 2019). One important implication of including data quality as a factor is that the technical debt of 2 implementations of the same DISS can be different, if their data is different, and identifying and treatment

of technical debt in DISS needs to consider the underlying data and its criticality (Weber et al., 2014).

Muse et al. investigated data access related technical debt and performance anti-patterns in a selection of SQL and NoSQL databases (Muse et al., 2022b,a). They investigated and categorised the self admitted technical debt (SATD) in data access related source code commits and proposed an 'data access' extension to Albarak and Bahsoon's taxonomy and downstream technical debt identification tools (Muse et al., 2022b).

The systems engineering discipline is also only beginning to apply and research the technical debt metaphor (Kleinwaks et al., 2023). The need to address data model debt in the systems engineering discipline was identified by Waltersdorfer et al. who developed a Production System framework that identified causes and mitigation approaches for new technical debt types. They incorporated a multidisciplinary perspective and developed a framework covering data model debt, knowledge representation debt and exchange process debt (Waltersdorfer et al., 2020).

2.3. Data-Intensive Technical Debt Management

Research on managing data intensive technical debt and how it differs from traditional technical debt management is still emerging. Muse et al. considered the time it takes to address SATD in data-intensive systems compared to traditional technical debt, and found that SATD remains active twice as long compared to traditional technical debt. They also found that bug fixing and refactoring were the main reasons for introducing SATD and that removal of data access SATDs was mostly associated with feature enhancements, new feature introduction and bug fixing, but not refactoring (Muse et al., 2022b). Tang et al. studied refactoring in open-source ML systems and created a Hierarchical ML specific refactoring taxonomy (Tang et al., 2021). The need to consider and balance data quality, refactoring and associated data migration costs motivated Albarak et al's development of a multi attribute decision analysis framework for prioritising database normalisation debt (Albarak et al., 2020; Albarak and Bahsoon, 2018). Sklavenitis and Kalles have proposed a methodology for quantification technical debt within competitive AI platforms. They consolidated and categorised existing technical debt research studies into technical debt types and factors relevant for measuring the technical debt type. Where possible they also identified mitigating strategies. Their methodology is under evaluation (Sklavenitis and Kalles, 2024).

2.4. Consideration of Practitioner Perspectives

Whilst empirical research about technical debt in DI systems is an active area of research (Tang et al., 2021; Muse et al., 2022b,a; Albarak et al., 2020; Waltersdorfer et al., 2020; Recupito et al., 2024), there is limited research on the practitioner perspective (Recupito et al., 2024). Some studies involved evaluations with developers (Aniche et al., 2014; Weber et al., 2014; Albarak et al., 2020), or wider set of participants (Waltersdorfer et al., 2020), whereas Recupito et al provide the first insights into state of the art practitioner perspectives on technical debt in the context of artificial intelligence systems (AITD). They selected a subset of hidden technical debt scenarios first articulated by Sculley at al. (Sculley et al., 2015) nearly 10 years ago. Surprisingly, Recupito et al found a low prevalence of AITD issues identified by practitioners, indicating that the state of practice of technical debt management for AI enabled systems is at a preliminary state and participants are not able to recognise issues in their systems (Recupito et al., 2024). Practitioners perspectives have been considered in software specific technical debt management practices (Freire et al., 2023; Guo et al., 2016; Xavier et al., 2023), but the extension of this work to multidisciplinary data-intensive software teams remains under researched.

2.5. Motivation for Our Study

Our previous exploratory interview study developed a theory about multidisciplinary data-intensive development teams having to deal with data challenges and identified strategies and approaches that teams use to address these (Graetsch et al., 2023). Whilst the findings of that study motivate the need for and chart directions for possible solutions, they lack detail about the actual work context on the practices of multidisciplinary data-intensive teams. The study did not capture (i) how key data-related challenge issues emerge in data-intensive software, (ii) how multidisciplinary teams identify, evaluate and resolve such issues, and (iii) what role their respective development tools and workflows play during team member interactions.

We wanted to take a human-centred approach to designing and developing solutions, but had insufficient information to drive use cases for solution development. A wide search of the literature about data practices/tools/multidisciplinary teams did not yield adequate detailed information. To address this, we developed a research agenda with the aim of filling this current gap in knowledge about multidisciplinary, data-intensive system teams and how they work, with a human-centred lens to focus on data work practices to:

- Contextualise and drive understanding of data-intensive delivery work practices
- Understand how experts in a multidisciplinary team work together on data scenarios
- Understand how they use their tools when working together

To achieve this aim we decided to adopt a fieldwork based case study approach in the form of a participant observation study using socio-technical grounded theory (Hoda, 2022) to analyse our data (See Section 3). We chose this approach over surveys and interviews to gain a greater depth of understanding of how a team works together on a project than is achievable through interviews or repository mining studies. Whilst conducting this 6 week observational field study, it became apparent that there were patterns of discussions and work by the team focused on identifying and managing data-related technical debt. Through our analysis, it became clear that this data-related technical debt was an important category of work done

by the team – a sizable portion i.e. more than 30% of the observed interactions referred to or discussed one or more technical debt or technical debt management concepts. We therefore wanted, in this paper, to analyse this large observational field study dataset to answer the following key research questions around the team's discussions and interactions about technical debt:

- RQ1: What does a multidisciplinary, data-intensive system engineering team discuss about technical debt?
- RQ2: How does the team identify and assess technical debt?
- RQ3: What does such as team discuss about technical debt treatment?
- RQ4: How does the team decide the treatment it will apply to the technical debt?

3. Research Method

3.1. Software Engineering Case Study

We selected the Case study research method as it supports empirical enquiry in real-life contexts; in particular, this method enables researchers to draw on multiple sources of evidence to investigate an instance of a "contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified" (Runeson et al., 2012). Runeson et al.'s Case Study method and guidelines have been specifically targeted to software engineering (Runeson et al., 2012; Wohlin, 2021). More recently, Wohlin suggested refinements to the definition of case study to focus on using multiple data collection methods, studying a contemporary (not historical) phenomenon in real-life (with people) context, with the investigator not taking an active role (Wohlin, 2021). Case Studies are most suited to answering questions of an exploratory nature and need to accept that the researcher has a low level of control over the study situation, in return for realism (Runeson et al., 2012).

Our aim was to study the phenomenon of a multidisciplinary data-intensive software team in situ, seeking a strong element of realism and detail, and the researcher was taking an observational role and not taking an active role in the study. Our research aim also aligned with Wohlin's more nuanced definition as we sought to investigate and develop an understanding of the practitioners' perspective and work practices i.e. how a multidisciplinary team of experts work together in a data-intensive software team.

3.2. Case Study Design

We followed the guidance of Runeson et al. regarding the elements of case research design. Based on the rationale and purpose set out in Section 2 above, we defined the object, or case, to be studied as an "exploration of a team of professional practitioners delivering a data-intensive solution", specifically the observable practices of the team members and how they inter-relate. This requires a broad perspective and as such we determined that a holistic design, where the case is the unit of analysis, would be most appropriate. Under this design, the context of the case consists of both the organisational context and the co-ordination (i.e. agile scrum) ceremonies.

The case study was designed such that the researcher should take an observational role only, with participants being fully informed and aware of the observations at all times. The researcher was seen as having the role of a researcher, and was not part of the delivery team. The aim was to observe the team members without interrupting their normal flow and to observe what actually happens.

The agile scrum ceremonies were key units of observation i.e. the researcher planned to attend as many team ceremonies as possible during the duration of the case study to observe team interactions. The design of our timeline had to balance our desire to observe end to end delivery, provide opportunity to observe reflective team meetings resulting in changes to practices (Dittrich et al., 2020) and respect the commercial realities of operating a team under a research environment. We estimated a timeline of 6 weeks based on the assumption that if the observed team worked in an agile delivery environment with 2-week sprints, it would allow observation of 3 sprints to provide observation opportunities for end to end delivery of features as well as provide insights into team reflection, continuous improvement efforts and in addition, the first author also planned to attend deep dive sessions to observe detailed interactions between team members. Whilst we planned to take an observer role only, we also planned for questions that may arise during observations about the context and practices and hence included clarification sessions with individual team members where we could ask questions in our case protocol.

Whilst observations were the main data source, we also planned to collect data about work items that the team members worked on, and interactions or messages that the team members exchanged. The case was designed for this data to be collected through observation notes, video recording interactions and access to messages being connected to the team collaboration system. To prepare and plan the observations, the first author reviewed literature about ethnographic observations (Sharp and Robinson, 2004; Sharp et al., 2016; Shah et al., 2014; Fetterman, 2019) and created templates and procedures for data collection (Spradley, 2016). She also completed self-paced observation exercises that focus on different aspects of observations to improve observation skills(Nippert-Eng, 2015).

3.2.1. Human Ethics, Privacy and Confidentiality

Whilst the need for more field work based case studies with human subjects in Software Engineering has been raised (Storey et al., 2020), these studies come with additional requirements regarding data collection and management. Entering the natural settings of participants, for a lengthy period of time, requires careful consideration of how to manage the interactions and any obligations that arise, and how to ensure reciprocity (Hammersley, 2020). These considerations need to be addressed throughout the research study, but come to fore first during the ethics

process. We followed the guidelines by Runeson et al. (Runeson et al., 2012) and our planned case study protocol was reviewed and approved by the Monash University Human Research and Ethics Committee (MUHREC).

The involvement of practitioners working in an organisation carries the additional requirement to protect commercially sensitive information. As we planned to video record and transcribe observations, we developed a protocol to collect data using the organisation's Zoom platform, and transfer the recordings to our university Google Drive. We committed to 'blurring' faces in the video recordings to protect participants' privacy, and blurring commercially sensitive data or information. We planned to have the voice recordings transcribed using Otter.ai and committed to removing the recordings from the platform at completion. We also committed to anonymising the Otter.ai transcripts. At conclusion of the data collection phase, all data would be stored in our secure university Google Drive, and our stored video recordings were blurred and our transcripts anonymised.

We developed recruitment fliers and emails. To support the requirement for 'informed consent', we prepared materials for different types of audiences. The case study protocol, including the data management protocol were summarised into a general 'recruitment presentation' and then more formally into separate Explanatory Statements for the organisation, team member participants and stakeholders that may attend meetings. We developed a pro-forma permission letter for the organisation, consent forms for team members and consent forms for stakeholders. We also created a "Zoom background" slide to be used by the researcher when attending Zoom calls. The background slide served to alert attendees that an observation was in progress and provided a QR code with links to the explanatory materials. We also created a 'flyer' for meeting room doors to inform any attendees who physically attend that the observation study was being conducted. We also committed to verbally notifying attendees that a recording would be taking place and seeking consent from all attendees prior to each meeting.

All materials were reviewed and approved by MUHREC. The aim of these materials and communications was to ensure that any attendee at the meeting was informed about the study and had the opportunity to provide consent prior to each observation and to request stopping the recording at any point. Further, our process had to ensure that we collected one formally completed consent form from each participant (team member or stakeholder) who participated in one or more observation meetings. We make these materials available for reference in supplementary materials.

3.2.2. Recruitment

The authors posted advertisements on LinkedIn as we wanted to attract professional industry participants, and also reached out to professional networks. We received a number of inquiries which proceeded to presenting our overview slides, of which one proceeded to engagement with the executive level of an education organisation. The first author was invited to present the study proposal to Data Capability Leads at the organisation. The Capability Leads then organised for the first author to present the study overview to the Date Engineering Group (DEG) executive committee. The executives had questions mainly pertaining to confidentiality and privacy and the first author articulated the elements in the protocol pertaining to privacy including anonymisation and data storage protocol. Subsequently, the first author was referred to the 'Star Squad'[1] project leaders - Iteration Manager and Product Manager to present the study overview and discuss the study and protocol in more detail.

3.2.3. Obtaining Consent

The project leaders raised the proposed study with team members without the presence of anybody from the research team to see whether the team members would be interested and whether they had any objections. The team members were interested in finding out more. The first author was then invited to present the overview of the study to the project team and provided the team members with copies of the Explanatory Statement and Consent Forms forwarded through the Iteration Manager. The team members had a number of questions about confidentiality and privacy - which were addressed through the data protocol and regular consent processes.

The first author was only provided with the names of team members and contact details as part of their completed consent form. Once all team members had provided their consent, the observation study was set to commence on January 30, 2024. The total recruitment duration for this study, from advertisement to commencement was 5 months. The duration from first contact with the organisation representative, including the various presentations and gathering of consent, until study commencement, was 3 months.

During the study, there were a number of observation meetings involving stakeholders. The sessions were recorded on Zoom as part of normal proceedings (as it was the Sprint Review), and once stakeholders had provided their formal consent, the recording was released to the first author.

Meeting Type	**Observations**	**Duration (hrs)**
Team Kick Off	1	3.0
Sprint Planning	4	10.0
Backlog Refinement	5	6.0
Daily StandUp	29	9.0
After Party	18	6.0
Deep Dive Session	11	5.5
Sprint Review	4	3.5
Sprint Retro	4	4.0
Clarification Session	15	7.5
Reflective Workshop	1	1.0
UAT Workshop	1	1.0
Total	93	56.0

Table 1: Summary of observation data

[1] Team and organisation name anonymousised at request

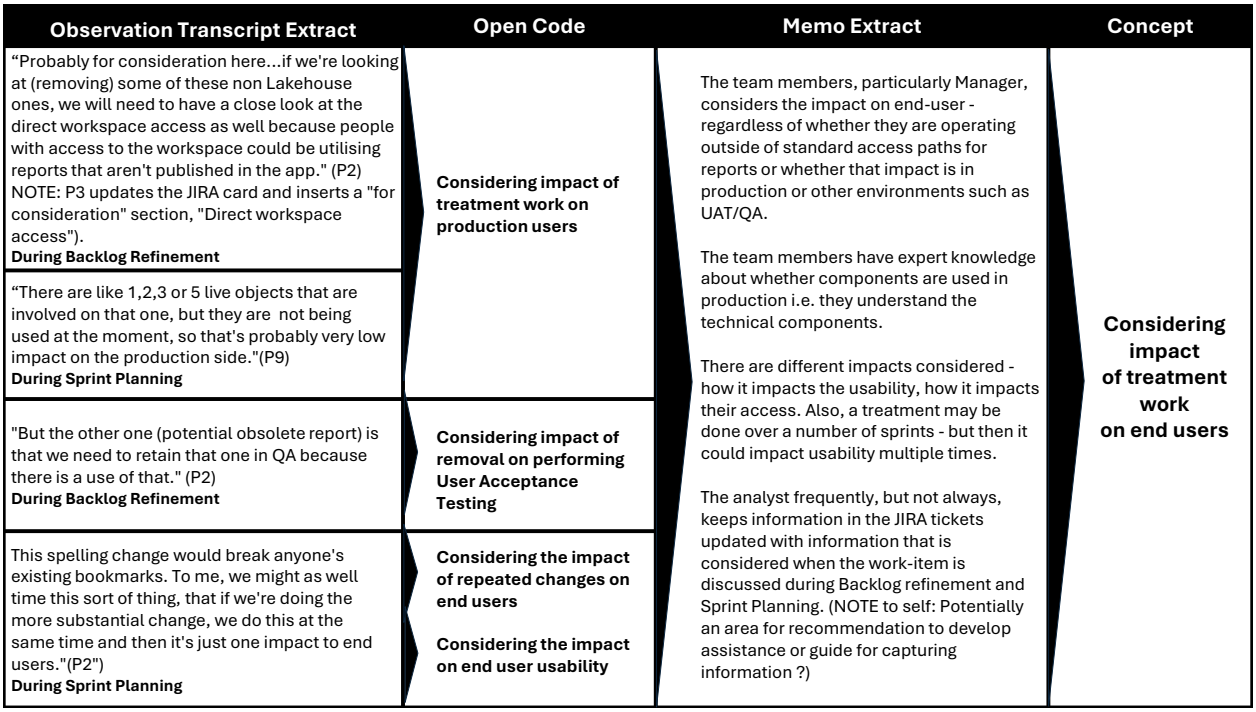

Figure 1: Applying STGT for data analysis: Example of open coding and memoing to develop a concept

3.3. Data Collection

The first author was 'onboarded' to the 'Star Squad' by being invited to all ceremonies - Sprint Planning, Backlog review, Daily Standup (and associated After Parties), Sprint Review and Sprint Retro. She was also included in the team's Slack [2] channel to be part of the day-to-day team collaboration messages. The initial data collection plan included an initial Workshop to capture context information. However, as the team had just recently been formed by merging 3 partial teams in an organisational restructure, the 'Star Squad' held a Team Kick-off workshop, facilitated by the Iteration Manager, and the first author proceeded to observe this meeting instead.

The team operated in hybrid mode and one or more team members were off-site for each meeting. Meetings were conducted as Zoom sessions. The first author 'attended' the Zoom meeting with her 'Zoom background' slide. Once participants provided consent, the Iteration Manager delegated control to manage recording to the first author (except for Sprint Review sessions). In addition to the scheduled ceremonies, the first author also attended additional Deep Dive sessions. These were selected based on discussions during daily stand-ups and covered a selection of analysis review discussions and release pipeline discussions. She made observational field notes during each observation, including information about the attendees at the session and any questions or issues arising out of discussions. She used these notes to formulate clarifying questions to be addressed with participants in subsequent Clarification Sessions, which were also recorded. The Clarification Sessions also provided an opportunity to capture education discipline and career background information from each participant. The observations commenced mid-sprint, with a Backlog Refinement session on January 30, 2024, and concluded with a Sprint Retrospective on March 20, 2024. The final reflective workshop was conducted on 22nd May 2024. We observed 93 sessions with a total duration of 56 hrs. A summary of the types, number of sessions and total duration of the observations is provided in Table 1.

3.4. Data Analysis using Socio-Technical Grounded Theory

Socio-Technical Grounded Theory (STGT) (Hoda, 2022) is an inductive, qualitative research method with guidelines that adapts traditional Grounded Theory methods from the sociology domain (Glaser and Strauss, 2017; Corbin, 2008; Charmaz, 2014) to the software engineering domain. STGT is specifically suited to socio-technical phenomena such as our exploration of a multidisciplinary data-intensive software team. STGT expects researchers to have a socio-technical focus and ability to understand the language and general technical domain. Our research team are experts in software engineering and have skills and interest in human-centric research. Further, the first author is an experienced industry project practitioner. The researchers had a good grasp of the observed team's technologies and domain and were able to understand the domain language, technology and processes that the participants referred to.

STGT caters for the analysis of data from a variety of sources such as observations, images and logs. It provides an iterative approach to collect, analyse and structure data into concepts, categories and theories. STGT starts with a broad topic and narrows analysis down to key phenomenon (in case of exploratory studies with no specific focus) or key categories. STGT has two

[2] https://slack.com

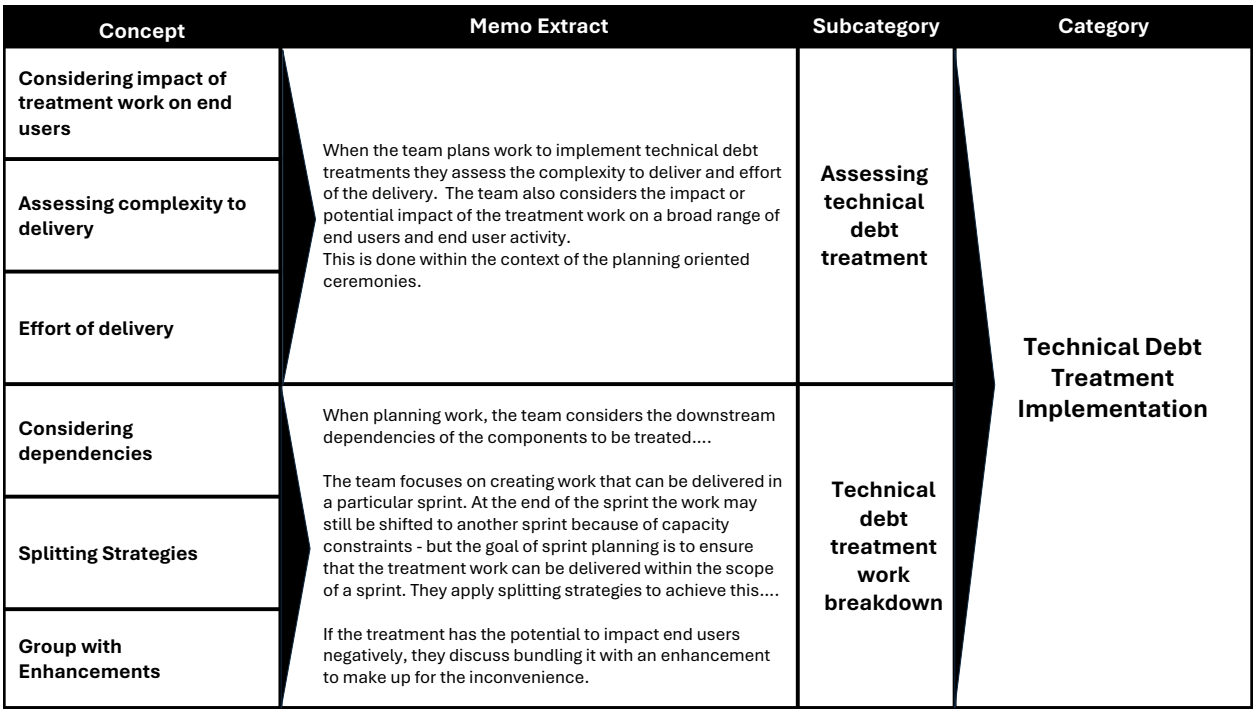

Figure 2: Applying STGT for data analysis: Examples of categorising concepts into subcategories and category

stages: a *basic stage* for basic data collection and analysis, and an *advanced stage* for mature theory development. It allows for limited application of the basic stage such as to analyse data collected using various methods. For this study, we used a limited application of STGT i.e. STGT *for data analysis* to analyse the data collected through the case study construct. During data analysis it became apparent that managing technical debt in a multidisciplinary DI software team was one of the key phenomenon of interest. Concepts and categories conceptualising the types and properties of technical debt, types of treatments, and work planning to treat technical debt emerged early in the analysis. We continued to focus our analysis on this phenomena by aiming to answer the research questions outlined in the motivation section.

The STGT method does not prescribe a singular research paradigm such as positivism, instead, it expects the researchers to articulate their perspective. The first author, who conducted the data collection, clarifying interviews and analysis, has extensive industry project delivery experience. We wish to acknowledge the adoption of a subjective, context specific, constructivist research paradigm needs to be acknowledged in constructing questions, interpretation of answers, formulation of concepts and categories and their interrelationships.

3.4.1. Preparing and Filtering the Data

Data preparation is "the process of converting the raw data into formats (typically text-based) where qualitative analysis can be performed efficiently" (Hoda, 2024, Chapter 9). Data preparation for analysis involved converting each Zoom recording into an annotated, anonymised transcript. We used Otter.ai[3] to transcribe the Zoom recording, converting it into text. We anonymised the transcript by replacing team member names, and references to named stakeholders and organisations, and we annotated the transcript with key references to the video, including the Jira[4] ticket that the discussion referred to, or descriptions about how participants interacted with tools as part of a discussion or demonstration. The aim was to ensure that key visual information from the observation was connected to the transcript to provide context during the analysis.

Per our agreement with the organisation, we also anonymised the video recordings. We downloaded the Zoom recordings to Adobe Premiere Pro[5], blurred identifying and commercially sensitive information and re-encoded the blurred video. We deleted the original. From the anonymised videos, we extracted relevant screenshots of Jira tickets (work items), Miro[6] boards, and Documents that had been shared and discussed during ceremonies per our case study design to enrich the analysis. We also extracted the team slack messages, ensuring that we captured each message and associated replies, at the end of each week into a spreadsheet. Per our agreement with the organisation, the anonymised videos and transcripts were made available to the participants (only the participants involved in the respective sessions), for a 2 week review period to provide participants the opportunity to raise any concerns, specifically regarding commercial sensitivity. Two participants provided feedback, thanking us for the opportunity to review.

One issue to highlight about data collection and anonymisation was the manual effort required. Whilst we used an au-

[3] https://otterai.com

[4] https://www.atlassian.com/software/jira

[5] https://www.adobe.com/au/creativecloud.html

[6] https://miro.com/

tomated transcription service, the associated manual effort was much higher than we anticipated and involved numerous corrections in addition to anonymisation. The meetings contained technical terminology, multiple speakers and a number of participants were not native English speakers. Every 1 hour of meeting required 2 hours of additional manual effort to correct and anonymise. Blurring of videos was also manually intensive, but necessary due to the privacy and commercial sensitivity of the material. Each hour of video required at least 1 - 2 hours of effort to anonymise, followed by encoding.

We imported the anonymised transcriptions and slacklogs into NVivo[7] for data filtering. Data filtering is "the process of identifying the key information, contextual information, and noise in the raw data" (Hoda, 2024, Chapter 9), to streamline later analysis of the data, decided that grouping by "work" was relevant information and so we elected to perform filtering with grouping. The first author reviewed the transcripts and used 'annotations' in NVivo to identify relevant segments, clarify meaning of passages and incorporate information from observation notes and Jira tickets, Miro board and Documents. We also commenced memo writing to capture reflections about key information - including organisational context information, data work practices, technical debt work and pipeline issues. We took screenshots of the Jira ticket from the anonymised video along with additional screenshots of document contents, diagrams or tool interactions. We created NVivo CASE records for each identified Jira ticket and linked annotated transcript segments and screenshots to the respective CASE record to facilitate a 'work based' case view, as well as a 'ceremony' file-based view. Segments filtered out included various discussions about team social activities and personal matters.

3.4.2. Data Analysis

To analyse the data we open-coded the technical debt related data consisting of filtered, annotated transcripts, screenshots and slack logs. We used the coding feature in NVivo (Hutchison et al., 2010; Soliman and Kan, 2004). Our analysis quickly identified that the observations of team ceremonies provided a rich dataset to surface technical debt characteristics and activities relating to how the technical debt was identified, and how solutions were selected and planned for implementation. We therefore established the key category of 'Managing Technical Debt in a Multidisciplinary Data-Intensive Software Team' to provide a focal point for our analysis.

We demonstrate the application of STGT basic data analysis with a worked example in Figures 1 and 2. When open coding, the first author sought to understand what was happening, what actions were taking place, and how those actions related to the key information identified during the filtering process. Additional segments were added to memos to capture reflections relating the codes to each other into concepts. A worked example of how open codes are assigned to build concepts through basic memos is shown in Figure 1. More conceptual memos where then developed to structure codes into subcategories and categories, as demonstrated in Figure 2. Diagramming was then used to visualise and further structure the categories and concepts. The first author regularly met with the others to discuss the analysis approach and review coding examples and progress. The results of the analysis in terms of identified concepts and categories are presented in the form of a theoretical framework shown in Figure 5 which shows the concepts, subcategories and categories identified within the overall case context. We also identify a concept called *evolution* on the boundaries between the case context and case to indicate signify ongoing and evolving impacts from the context to the unit of analysis. The case context and detailed conceptual findings (including the concept of evolution) is explained in the next Section.

4. Findings

4.1. Case Study Context

4.1.1. Organisation Structures

The observed 'Star Squad' was responsible for the delivery and support of the Enterprise Reporting system. The observed team belonged to the Data Engineering Group (DEG) in the organisation's broader IT Group. The observed team was one of 8 teams in DEG, and there were 12 team members out of a total of 73 across the 8 teams. The observed team members and roles are outlined in Table 2. Team members were allocated to capability groups and community of practices (COPs) aligned with their roles and interests. For example, analysts in the observed team members belonged to one of the Analytics and Design, Data Engineering, or Ingestion capability groups. Within the Analytics and Design, there were three COPs - Data Analytics and Design, Visualisation, and Advanced Analytics. The operation of the other teams, COPs, and Capability groups were not included in the observed scope, but are included for context. See Figure 3 for an overview of the organisation structure.

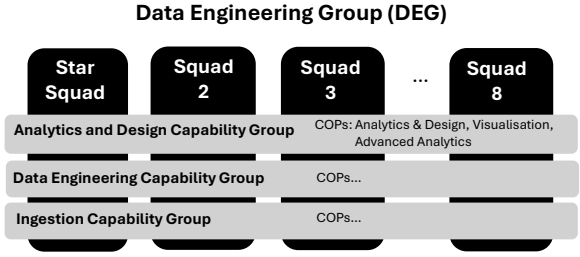

Figure 3: Context: Star Squad Organisation Context

4.1.2. Product and Stakeholders

The enterprise reporting end product consisted of a number of PowerBI[8] dashboards deployed through workspaces, as well as access to the underlying datasets (available to power users) and personalisations (available to all PowerBI users. The

[7] https://lumivero.com/mylumivero/

[8] https://www.microsoft.com/en-us/power-platform/products/power-bi

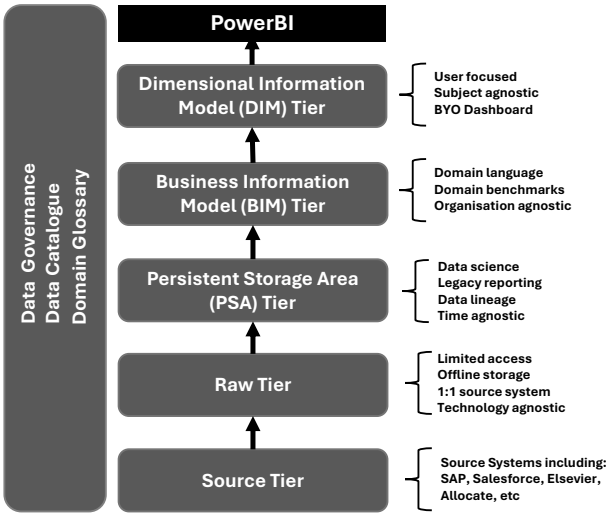

Figure 4: Context: (Anonymised) Overview of Organisation Datawarehouse Architecture Tiers

end user community was made up of data analysts as well as senior management and executive stakeholders. Data analysts who had different levels of analytical skills had access to the dashboards and datasets, whereas management level stakeholders generally had access to dashboards and perhaps personalisation features. End users, mainly data analyst end users, were connected to the observed team through COPs and also engaged as 'domain experts' in specialised groups for consultation and user acceptance testing. The team also regularly engaged with a data governor.

4.1.3. Technology, Architecture and Tools

The observed team works in a Business Intelligence Enterprise Reporting Datawarehouse environment (Kimball et al., 2008). DEG is in the process of implementing the Databricks[9] Lakehouse Platform. This platform enables a tiered Data Warehouse Architecture as shows in Figure 4, which connects multiple source systems through the base tiers responsible for processing the raw data (Raw Tier) and separating out personally identifiable information from other information into the Persistent Storage Area (PSA Tier). These lower level tiers are outside the scope of our study as they are developed and maintained by other DEG teams. The Business Information Model (BIM) Tier provides models set up in business language. It feeds central dashboards, whereas the Dimensional Information Model (DIM) provides the final Tier to support end user accessible dashboards and datasets accessible through Microsoft PowerBI models and visualisations. The organisation has established Data Governance processes as well as Data Catalogues and Business Glossary. The details and operation of these processes and structures were not included in the case study observation but are noted here for context.

The BIM and DIM layers in the Lakehouse Architecture had been developed incrementally over a period of 18 months. The development teams had recently been restructured into the structure of 8 teams noted in Section 4.1.1, including the 'Star Squad', which was the subject of observation.

We observed the 'Star Squad' members use several tools, including Databricks Catalogues and Notebooks for SQL development (BIM and DIM), Microsoft PowerBI for data visualisation and dashboard development, and Azure DevOps Pipelines[10] for BIM and DIM workflow processing and release deployments. PowerBI Deployment pipelines are used for PowerBI deployments. Importantly, the Azure DevOps Pipelines and PowerBI pipelines were not yet integrated due to technical limitations, which contributed to challenges elaborated in Section 4.2 below. The team used Lucidchart[11] for data modeling and Atlassian Confluence[12] to document BIM designs. The team used Atlassian JIRA[13] to track, manage, and report work in sprints. Outside of team ceremonies, day-to-day team collaboration was done through Slack[14], and team ceremonies were always in hybrid mode, conducted over Zoom[15]. At times, meetings were facilitated through Miro and associated templates[16]. The team also had access to Google Workspace tools[17].

4.1.4. Team Members

The Star Squad was a multidisciplinary team, consisting of 8 data analysts, a product manager, iteration manager, business analyst, and quality assurance expert, as shown in Table 2. The team members' education discipline backgrounds are shown in Table 3, and their career backgrounds are summarised in Table 5. Some team members had education backgrounds in more than one discipline. The discipline backgrounds span a wide range of domains, including non technology related domains, general information technology, engineering, data science, computer science data analytics, and management. The Product Manager, in particular, had established deep domain knowledge in the organisations domain, with most of their professional experienced gained working in similar organisations. They also had a deep knowledge of the organisation's legacy application environment. Analyst team members were largely aligned to tasks relating to visualisations, or backend data analysis and query development. The team had a balance of very deep level of professional expertise and junior team members (See Table 4). Note, the information in team members related Tables 2, 3, 4, and 5, has been generalised and aggregated to preserve team member privacy.

4.1.5. Ways of Working

Ceremonies: The 12 member team used agile scrum practices to manage their work. They held the following ceremonies (with time ranges) fortnightly: a) *backlog refinement* (1hr -

[9]https://www.databricks.com/

[10]https://azure.microsoft.com/en-us/products/devops/pipelines/
[11]https://www.lucidchart.com/pages/
[12]https://www.atlassian.com/software/confluence
[13]https://www.atlassian.com/software/jira
[14]https://slack.com/intl/en-au/
[15]https://zoom.us/
[16]https://miro.com/
[17]https://workspace.google.com/intl/en-au/

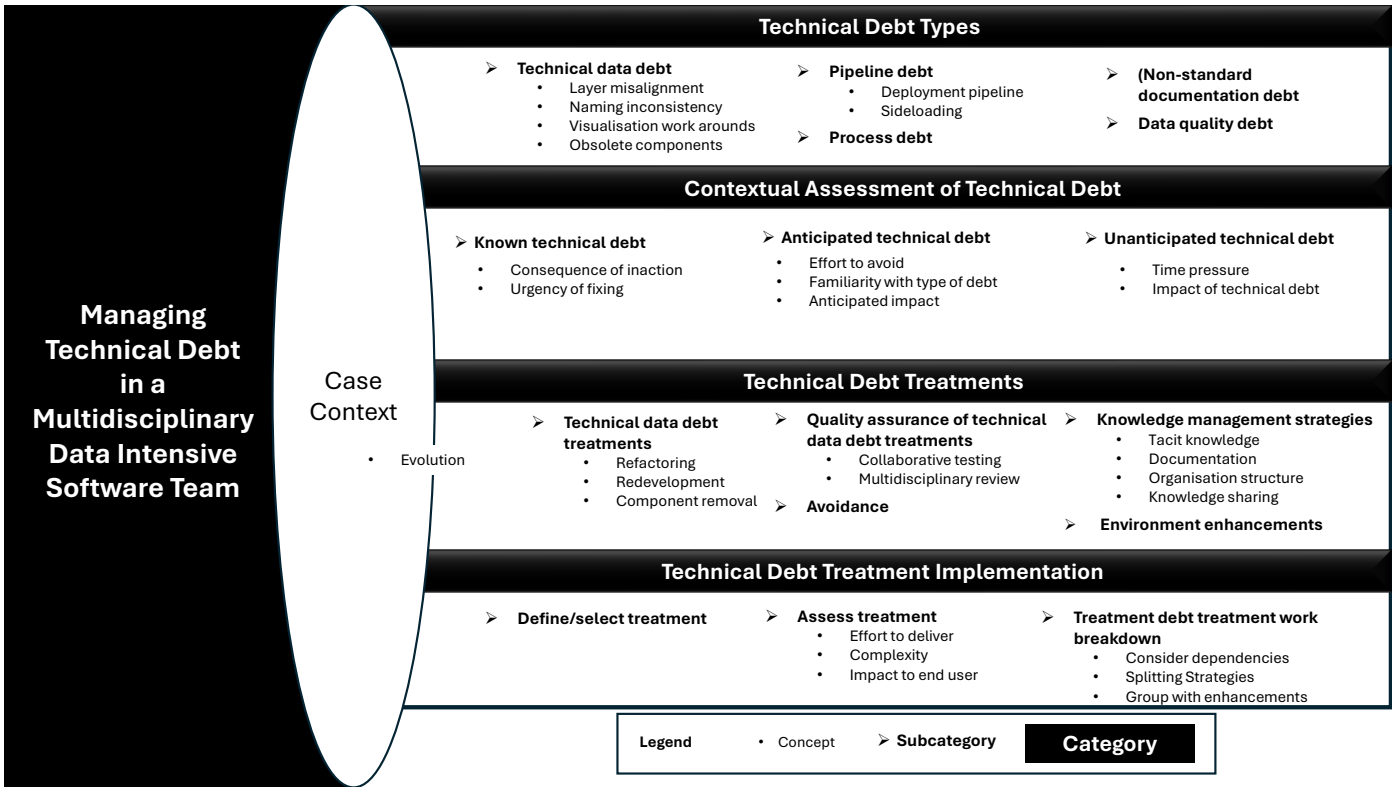

Figure 5: Theoretical framework of Managing Technical Debt in a Data-Intensive Software Team

ID	Role (Focus Area)
P1	Manager
P2	Manager
P3	Analyst
P4	Data Analyst - Backend
P5	Data Analyst - Platform
P6	Data Analyst - Visualisation
P7	Data Analyst - Visualisation
P8	Data Analyst - Backend
P9	Data Analyst - Backend
P10	Data Analyst
P11	Analyst
P12	Data Analyst - Visualisation

Table 2: Participant Team Member Roles

Education Discipline	No. of Team Members
Domain	2
Information Technology	2
Data Analytics	2
Engineering	3
Computer Science	2
Data Science	2
Training	1
Business Management	1

Table 3: Aggregated Participant Team Member Education Disciplines

Years of Experience	No. of Team Members
< 2	2
2 - 4	0
5-10	5
>10	5

Table 4: Aggregated Participant Team Member Years Professional Experience

1.5hrs) to scope out, estimate and prioritise work items, b) *sprint planning* (3hrs) to close out the existing sprint or carry over work from the prior sprint and finalise (scope, estimate and prioritise) the items for the commencing sprint, c) *sprint review* (30min-1hr) to review the previous sprint and showcase deliverables within the team and to external stakeholders, and d) *sprint retrospective* (1hr) to review sprint performance and conduct reflection on how to improve. Every day, except on sprint-planning days, the team held a *stand-up* (10min - 30min) ceremony to discuss progress and, if required, raised the need for an *after-party* (10min - 50min) ceremony to discuss issues, next steps, or conduct reviews with each other and/or the product manager. All ceremonies were attended by all team members, except after-parties, which were held on most days and only attended by those team members involved in the discussion.

Observed Data-Intensive Work: For the purpose of contextualising, we characterise the observed DI work performed by team members as: a) *data development*, b) *data visualisation development*, c) *quality assurance*, d) *data deployment*, and e) *Visualisation deployment* work to deliver the enterprise business intelligence product. For the purpose of this paper, we define *data development* work as including the SQL and related coding work including the design, creation and updating of Business Information Models (BIM) and Dimensional Data

Career Backgrounds	No. of Team Members
Business Analysis	2
Data Engineering	5
Data Modeling	2
Data Reporting and Visualisation	3
IT Training and Support	2
Product/Project Management	3
Quality Assurance	1

Table 5: Aggregated Participant Team Member Career Backgrounds

Model (DIM) data tiers in the architecture. We define *data visualisation development* work as the data modeling and visualisation development in PowerBI. Quality assurance work activities included testing and review activities associated with *data development*, *data visualisation development* and their integration. *Data deployment* work includes the maintenance and continuous updating of workflows and pipelines to accommodate changes in the BIM and DIM data into the Development (DEV), Quality Assurance (QA) and Production (PROD) environments. *Visualisation deployment* includes the deployment of PowerBI dashboards and refreshing of data into the data visualisations in the DEV, QA and PROD environments.

Team members were observed to perform or discuss the performances of DI work shown in Table 6.

Measured Work Types: DEG teams were measured through various metrics discussed at leadership meetings (outside the scope of the observation study). The metrics were collected through Jira work item types, supplemented by highlight descriptions provided by the product manager and iteration manager. For the context of the findings related to technical debt, relevant information includes that the measured 'Work Item Type' and % range of work completed each month over the prior 4 months included *Break Fix:* <5%, *continuous Improvement:* 8% - 9% , *discovery incl. analysis: 0 - 9%* and *value work: 80%- 90%* . Further, the amount of *continuous improvement* work was kept relatively stable, ranging between 8% - 9.5% per month, closely related to and balanced with value work. Continuous improvement included enhancement work items as well as technical debt work items. The team was not observed to have a formal register for maintaining technical debt.

4.2. Managing Technical Debt in a Data-Intensive Software Team

We conceptualised several categories and concepts around our key category of **Managing Technical Debt in a Data-Intensive Software Team**, which we present in this section. The category **Technical Debt Types** is made up of sub-categories that describe the **Technical Debt Types**. We identified 3 further categories which conceptualise activities relating to technical debt management, namely **Identify and Assess Technical Debt**, **Technical Debt Treatments** and **Technical Debt Treatment Work Breakdown**.

Note: We deliberately omit the use of financial technical debt language such as 'debt repayment', and 'interest' when conceptualising. Instead we used language such as'technical debt treatment' and 'technical debt treatment implementation' as we felt the observed concepts, grounded in observation data, are broader than concepts such as 'debt repayment'. This approach also serves to highlight that financial technical debt language, beyond phrases such as 'technical debt' were not used by the team during the observations.

For each category, we present related sub-categories and concepts, some illustrated by sanitised quotes and conversational extracts. NOTE: We use –> in the following subsections to indicate where a quote continues the conversation. Figure 5 provides a visualisation of this conceptualisation. We present a summary at the end of each category in answer our research questions.

4.2.1. Technical Debt Types

We observed several references and discussions relating to technical debt that we separated into **Technical Debt Types**, namely **technical data debt**, **pipeline debt**, **process debt**, **data quality debt** and **(non-standard) documentation debt** sub-categories.

Technical data debt encompasses debt relating to BIM, DIM and PowerBI *layer alignment* where data is accessed directly from BIM structures (because the DIM layer has not been created), or the data layers still contain references to legacy components. The concept also includes *naming inconsistency* covering inconsistent naming data fields or report naming conventions, and DIM naming issues. For example:

> *"to be able to do this, we would be referencing BIM tables...and there is a need to assess the Tech Debt that we would create, by doing this piece directly off the BIM"* (P2, Manager).

We also include concepts such as *obsolete components* and Power BI *visualisation work arounds* within this sub-category. *Visualisation workarounds* included the need to implement modifications within PowerBI to achieve desired visuals due to underlying data structure or layer issues. An important impact of implementing modifications within the PowerBI visualisation layer is that access restrictions may need to be placed on access to the underlying dataset, impacting dataset access or personalisation by end users.

Within the sub-category of *pipeline debt*, we identified 2 main concepts - *deployment pipeline* and *sideloading*. *Deployment pipeline* related discussions centred mainly around issues with the deployment pipelines and data refreshes for PowerBI, which were not integrated with the Databricks Azure DevOps deployment pipelines. Workflows to 'build' data constructs ran on an automated schedule within the Azure DevOps environment, but PowerBI model refreshes were scheduled or executed ad-hoc through the PowerBI Service. Due to technical platform limitations, integration of these pipelines could not be automated, and running selective Azure Ops orchestrations without PowerBI refresh caused downstream data alignment issues that resulted in outdated 'keys' joining the FACT and DIM tables in datasets, which in turn resulted in misaligned datasets and nonsensical data flowing through to PowerBI dashboard visulisations. The team had implemented a manual co-ordination pro-

ID	Data Development	Data Visualisation	Quality Assurance	Data Deployment	Visualisation Deployment
P2	●		●		
P4	●		●	●	
P5	●		●		
P6		●	●		●
P7	●	●	●	●	●
P8	●		●	●	
P9	●		●	●	
P10	●	●	●		●
P11			●	●	●
P12		●	●		●

Table 6: DI Work performance observed or discussed by team members

cess via the team slack channel to notify team members prior to running Azure Workflows, and this would then alert team members developing visualisations that they would need to refresh:

💬 "*I'm about to kick-off dev qa orchestration. Please let me know if you have any concerns. Thanks!...Orchestration completed.*" (P8, DA-Backend)

Over time, the approach to deployment pipelines for applications had changed, leaving different approaches for older applications.

💬 *"Do we have deployment pipelines for those reports?"* (P2, Manager)
–>*"Not consistent, that was something that we implemented when we first started."* (P7, DA-Visualisation)
–>*"In the release meeting (with the other teams) it was mentioned that when we release some of these reports, we don't have a pipeline."* (P12, DA-Visualisation)

Whilst most data consumed by the visualisations was sourced through source systems, the Lakehouse architecture also supported *sideloading* of data files created through external (manual processes) necessary to fill gaps in source systems. *Sideloading* is a form of **pipeline debt** because the associated pipeline was identified as not having support for roll-back or archiving. The reason for requiring *sideloading* is that historically, the enterprise reporting team took on responsibility for managing some reference data on behalf of business stakeholders using Excel files, and those processes never progressed through digital transformation:

💬 "*I had a look at reference sheet. You mentioned about getting back to the 'previous version'. So, if we have to get back to a previous version, because it has multiple tabs, will it be the case that we have to revert back all of those references to the previous version.*" (P8, DA - Backend)

As source systems are updated and their data is integrated to the Lakehouse, effort needs to be expended to assess whether the *sideloading* can be replaced with direct access:

💬 *"So rather than us manually uploading that as a .csv file into our platform, like we used to do, we might be able to source this from < source system > instead."* (P2, Manager)

The team had practices for incorporating documentation for analysis, design and testing. However, there was missing documentation for some non-standard, legacy applications and pipelines i.e. **(non-standard) documentation debt** was identified and needed to be addressed when one of the team members announced they were leaving to take up a new job opportunity. **Process debt** related discussions were observed regarding processes and features for managing communication to end user stakeholders about data related issues:

💬 *"We've taken a few different approaches with some of the dashboards (for communicating with end users), there's different levels. There's removing access to the dashboard and replacing it with the Service Desk message. Then we've got that 'news box' on the landing page (for one of the dashboards), and on the (other dashboard) we just put text directly on to that landing page to communicate things previously as well. So we don't have a real consistency with our production workspaces at this point.*" (P1, Manager)

The team also discussed **data quality debt**, mainly in relation to historical data from older systems.

Summary

RQ1: What does a multidisciplinary, data-intensive system engineering team discuss about technical debt? We identified discussions about **technical data debt**, **data quality debt pipeline debt**, **process debt** and **(non standard) documentation debt** sub-categories of technical debt. The team members articulated that **process debt**, and **(non standard documentation debt** resulted from shortcuts taken during development. However **technical data debt**, **pipeline** and **data quality debt** were experienced due to the nature of data, or combination of data and technology. **Pipeline debt** in particular was discussed in terms of, and managed through, manual processes.

4.2.2. Identify and Assess Technical Debt

As part of their work delivery and planning, we observed that the team regularly identified, discussed and assessed technical debt. We established 3 subcategories based on the level of awareness or anticipation of technical debt, specifically **known technical debt** (to represent debt technical debt that had been identified and was known to the team - either documented as a work item or tacit knowledge), **anticipated technical debt** and **unanticipated technical debt**. Assessment of **known tech-**

nical debt and **anticipated technical debt** occurred mainly during fortnightly sprint planning and backlog refinement ceremonies, whereas **assessment of unanticipated technical debt** occurred during the sprint delivery in either daily stand-ups, after-parties or other deep dive sessions.

When the team assessed **known technical debt** they considered *urgency*, specifically how urgent it would be to treat the debt to achieve the current release, or to remove links to legacy components that were planned for decommissioning. The team members considered dependencies of other components, and *urgency* in terms of short term consequences.

💬*"With regards to that 'universal day' (Item identified for removal). We might need to just understand if there has been used elsewhere as well, before we do anything with it. Like we have used date dimensions in everything. So if they're all using this, and I mean if it is legacy, then we'll need to make sure that we address all of that before we go on remove that kind of thing.*" (P7, DA - Visualisation)
–>"*leaving it there as it is, like, doing nothing. There's no risks in the short term.*" (P2, Manager)

When assessing **anticipated technical debt**, the team members were observed providing their respective expertise and discussing and balancing factors including *familiarity with the type of debt*, *anticipated impact* if the debt was incurred, such as *anticipated impact* on system performance:

💬*"We would be referencing the BIM tables and there's an assessment around the Tech Debt that we would create by doing this piece of work directly off the BIM. Once we do eventually create the DIM layer, we would ideally look to refactor that to be consistent and point to the DIM. Particularly this BIM is more like a FACT table already... It's not the same as a collection of raw tables that we're bringing together for the BIM. P9, I'm looking to you for feedback about what you needed to do previously, was that relatively straightforward to use the BIM directly for that?"* (P2, Manager)

–>"*I don't see any difference between the logic that we already developed. The only drawback with this one is that we have to maintain the same filter on this work as well, because we don't have like a single component which filters out criteria, if that makes sense.*" (P9, DA-Backend)

In general, at conclusion of a discussion, the Manager summarised and articulated the decision, seeking to balance the anticipated impact of the technical debt against the estimated *effort to avoid* the technical debt and *anticipated impact* on sprint goals and effort to apply future treatment given *familiarity with the type of debt*:

💬*"If we wait to fully develop out the dimensional layer...it would add quite a large dependency or prerequisite, before being able to do our summarized measures. And initially, I'm proposing what we take the same approach with this one. So we've got that sort of equivalent, tech debt on both of those solutions...and take the slightly more tactical approach, particularly because it's just a simple flat table, and it is easy to replumb to a different source later...my appetite is to not take on too much of this within the one sprint."* (P2, Manager)

We also observed decisions about **unanticipated technical debt** during deep-dive sessions, prior to releasing the dashboard. One decision was made under time pressure of release, the other did not have pressure to release. Under this *real time pressure*, the main factor voiced by the Manager in making the decision was *end user usability*. Completion of the work involved additional, but not extensive effort by the analyst team member, but this was not considered a major concern:

💬*"The next things all really relate to fields names. And this is sort of both what you see in the filter panel, but also when you go to personalize visuals. It may end up being confusing (for end users, so that would say to me that we probably should put a prefix or a suffix on these names to make it clear."* (P2, Manager)

💬"*What I've done (since yesterday), ...I have prefixed it ...in the Power BI. We made a backlog card to fix these(in the VIEW).*" (P12, DA - Visualisation)

–>"*If that's the pathway we need to go down in the shorter term versus the VIEW, then ok.*" (P2, Manager)

By comparison, where the team made an unanticipated identification of data and technical debt during a review session, without strong release pressure, the assessment resulted in the decision to refactor of the query (and carry work to the next sprintto ensure the technical debt was addressed prior to release:

💬*"I think we'll need to make a call at the end of the day. And that if there's still a bit of work, there's no outward pressure to get this one released. So if we need a bit more time...I think the important step is to just remove that legacy query."* (P2)

Summary

RQ2: How does the team identify and assess technical debt?

Identification and Assessment of Technical Debt is contextual. In the case of **known technical debt**, the team considered the *consequence of inaction* and *urgeny of fixing*. However, if **unanticipated technical debt** is identified i.e., arises unexpectedly during a sprint, then the team considered the *impact of the technical debt* and the *time pressure* of delivery. They distinguished between 'real' delivery time pressures compared to 'sprint' imposed performance time pressures. The team chose to 'carry over' scope to fix technical debt when there were no real perceived time pressures. Finally, during planning, the team identified potential or **anticipated technical debt** and considered their *familiarity of the type of debt*, the *effort to avoid* and *anticipated impact* of the technical debt.

4.2.3. Technical Debt Treatment

Our observations identified the application and consideration of different types of **Technical Debt Treatments**, as well as the emergence or *evolution* of treatments. The **Technical Debt Treatments** category includes the **technical data debt treatments** sub-category with *refactoring*, *redevelopment*, and

component removal concepts. We also identified an associated **technical data debt treatment quality assurance** subcategory which includes quality assurance concepts such as *collaborative testing*, *multidisciplinary reviews* and *regression testing* to ensure that the **technical data debt treatments** are carried out as expected without adverse or unexpected impacts.

When discussing *refactoring*, the team considered updating field names, BIM and Report names, *refactoring* SQL Queries to reference newly created BIMs. *Component removal* treatment activities included the removal of obsolete SQL notebooks, PowerBI dashboards and PowerBI workspaces no longer required. We identified a relationship between *component removal*, *refactoring* and **treatment quality assurance** because refactoring activities that touch obsolete components need to be completed before *component removal* followed by *collaborative testing*:

💬 *"We identified some refactoring might be required. So in terms of like the actual removal of the BIM and DIMs is probably not complex, but the refactoring before and regression testing after - ensuring that we still produce what we are expecting."* (P8, DA - Backend)

We observed discussions about *redevelopment* to treat technical debt. The reasons for *redevelopment* (for 2 of the components) were that their existing implementations could not be extended to accommodate new requirements. The other *redevelopment* was planned to consolidate a number of different approaches that had been implemented on dashboards to communicate with end-users:

💬*"I've asked P3 to raise an item in the backlog, to look for a more effective ways to raise the service messages, that reduce some of the overhead."* (P1, Manager)

–>"*This work is coming under our continuous improvement area.*" (P2, Manager)

The team discussed and was observed performing **technical data debt quality assurance**, including *collaborative testing* and *multidisciplinary reviews*. Team members incorporated acceptance testing criteria when planning **technical data debt treatment** work. They performed *collaborative testing* where **technical data debt treatments** spanned multiple layers of a solution. For example, regression testing performed by P9 after the removal of obsolete notebooks required refresh of downstream PowerBI models by team members responsible for data visualisation, within the testing environment to verify that there was no impact. Different team members responsible for each of the PowerBI visualisations performed the tests. *Refactoring* updates to BIM/DIMs, performed by backend data experts also required refreshes and *collaborative testing* in the downstream PowerBI model by visualisation experts:

💬"*I removed all the notebooks and tables or views that's not been used or are inactive....I tested in DEV and it looks all good. I requested to P12 to refresh QA to make sure that there's no impact on the Power BI data set on (the dashboards P12 is responsible for) and it looks okay. Now I need help from P6 to refresh the dashboards (that P6 works on) just to make sure that it's not going to be impacting those two as well. And once that is confirmed then I'll get P11 to review if any further testing is required.*" (P9, DA - Backend)

The team used **knowledge management strategies** including holding *knowledge sharing* sessions and *documentation* of non-standard practices and implementation. This particular strategy was implemented one of the team members was planning to leave the organisation and resulted in knowledge being handed over in several sessions to 2 team members:

Knowledge sharing in the form of creating, or updating confluence documentation and hand-over meetings was used to address gaps in team member knowledge as a result of one of the team members taking up a new role with another organisation. The team member had responsibility for supporting a number of non-standard reports and pipelines and hence the team needed to ensure that they could keep up with the continued support. Whilst the team has some reservations about including this as a 'tech debt' related activity, they chose to group it as such for planning and reporting purposes:

💬 *"Basically on this one, I'm working on Confluence. One page for each project or dashboard, including the one I am working on at the moment...I am also updating existing documentation which is still, you know, referring to legacy."* (P6, DA - Visualisation)

💬"*That's the most sort of critical one, obviously, while you're here you know, capturing as much detail and those knowledge sharing opportunities while you're here.*" (P2, Manager)

We also observed how *organisation structure* facilitated *knowledge sharing*. P1 took on a Management role in another team that provided platform services to the 'Star Squad' and this facilitated *knowledge sharing* during that team's ceremonies. P1 indicated that this was a key factor in facilitating common understanding and improvement of managing *deployment pipeline* debt:

💬*"It was good to have me now also in the role of Manager with the <(other) team> - it helps with the alignment. We have 'common language' when we mention branches and releases...I notice in stand ups, the language seems to just sit. Now, when people talk about 'feature branches' we know exactly what they're talking about...so it seems to have landed with everyone".* (P1, Manager)

Environment enhancements including *process enhancements* were used to address *deployment pipeline* debt. However, the team was unable to fix the technical integration issues relating to the underlying *deployment pipeline*, and instead implemented *process improvements* to address the impact caused by *deployment pipeline*. The team is also actively monitoring availability of *technology enhancements* to address improvements with the PowerBI pipeline:

💬*"One thing we're doing extra now is a final QA master branch orchestration power Power BI refresh. If it works, it's almost guaranteed success. And that's something that has been introduced over the last few weeks. So after our release meeting, we gather up all of the disparate branches from all the*

teams and we perform a review of the final branch. Then we run the orchestration from start to finish and we do a Power BI refresh. Then we can run a regression test." (P11, Analyst)

We identified *evolution* as a relationship of the **Technical Debt Treatments** category, which provides connection between the treatments available to the team and the wider context of the case. Treatments may *evolve* due to work (carried out by team members and others) as part of data governance activities, Community of Practice activities, or may become available due to wider organisational activities or vendor product releases. This evolution operates at a different cadence to sprints:

💬*"There are multiple Communities of Practice...which run with all team members from the DEG...and we look at naming conventions, BIM and DIM data design practices, visualisation standards....The Community of Practice works at a different speed as our 'Star Squad' and involves different people.*" (P9, DA - Backend)

The *evolution* of technical debt treatment is not under the full-control of the team and needs to be evaluated. Once new technology becomes available, the team needs to investigate whether it could be useful and add value:

💬"*We should look at "Power BI files and Git repos - BIP files.*" (P7, DA - Visualisation)
–>"*They make it easy to track changes (in PowerBI) so you can avoid deploying something you don't want to go.*" (P11, Analyst)

Finally, we also observed **avoidance** as a treatment approach for *anticipated technical debt* as the team would try to find ways to de-scope the cause of technical debt. For example, old **data quality debt** from years prior to 2000 would be identified during sprint-planning, and the team took decisions to actively descope the data from the feature and **avoid** incurring the debt.

Summary

RQ3:What does such as team discuss about technical debt treatment?
We identified **technical data debt treatments** including *refactoring*, *redevelopment* and *component removal* as well as associated **technical data debt treatment quality assurance**. Multidisciplinary teams rely on *collaborative testing* and *multidisciplinary reviews* where different team members are called to contribute expert knowledge and perform **technical data debt treatment quality assurance**. We also observed the team discuss efforts to **avoid** technical debt, apply the **knowledge management strategies** and make **technical environment enhancements**. Available treatments *evolve* in line with the team's internal improvement processes, but are also facilitated and influenced by connection with the team's wider context such as COPs, Data Governance and Vendors. External sources of *evolution* do not align with cadence in agile sprints.

4.2.4. Technical Debt Treatment Implementation

We observed several discussions during the team's sprint planning and backlog refinement sessions in which the team defined and refined the way that treatment should be implemented. We conceptualised these findings into the **Technical Debt Treatment Implementation** category, which is made up of related sub-categories **define treatment**, **assess treatment**, and **technical debt treatment work breakdown**.

Define treatment could be as simple as raising one or more backlog tickets during review meetings to capture or identify the **Technical Debt Treatment**. However, we also observed discussion about the need to allocate time to think through and design appropriate treatment solutions, or even decide whether a treatment was possible:

💬*"So we do need to spend a bit of time to really understand the requirements around all of this reference data configuration data. And obviously, I talked about how we managed that in Legacy, which was just in an Excel spreadsheet, where we then exported each worksheet as a separate CSV file to load into the platform. But it'll be worthwhile for someone to have a look through that and propose other options for how we might want to manage that, obviously, it's not ideal managing such critical data in a in a sort of manual form like that, like mistakes can be made. They can have significant impacts to the to the reports and data. A review of what we need to manage in that space will be worthwhile as well."* (P2, Manager)

The **assess treatment** category has an implied precondition that there is a **Technical Debt Treatment** available i.e., that **define treatment** work has been done. We observed the team **assess treatment** for the purpose of planning implementation. The team discussed the treatment and specifically considered *effort to deliver* the treatment, *complexity of the treatment* and potential *impact to end users*. With respect to *impact on end users*, this included consideration of potential negative and critical impacts of the treatment, in particular the timing given the current point in the business cycle:

💬*"And this is the stuff that was considered medium priority last week. Should we just consider like how critical it is to do these? I think (previous technical team member) would have said, 'it's quite critical', but P2-PM has a different viewpoint. Especially with these particular FACT tables, I'll be honest, I don't really want to mess with it midway through this critical period (for end users) - I'm not saying 'Get rid of it'. But some of those other FACT tables, like, no one uses them, I reckon that's safe."* (P11, Analyst)

💬*"So if we only implement this spelling change, it will break anyone's existing bookmarks."* (P2)

We also observed the team *breaking down technical debt treatment work* with the goal of ensuring that the treatment could be delivered within the boundaries of the fortnightly sprint. When scoping and structuring the work, we observed the team *consider dependencies* and apply *splitting strategies* and *grouping with enhancements*:

💬*"So regarding the ...FACT, we do have the renaming of the FACT included in the refactoring. I'm assuming that renaming is not part of this task, because then that will impact the Power BI report?"* (P12, DA - Visualisation)

The team used *splitting strategies* to structure the work for delivery. Splitting strategies consider splitting by domain, dependencies, complexity or work impact:

💬*"Potentially we can split by subject area, for example, like we can do like the DIMs and FACT related to <domain area> and the second one will be like the <domain area>."* (P8, DA - Backend)

–>*"If you look at it, there are 12 there, right? So if we consider there is effort on this from a refactor regression. If you lump it all in one, we could be sitting on it for quite some time."*" (P1, Manager)

Summary

RQ4: How does the team decide the treatment it will apply to the technical debt?

The team applied steps to select or **define treatment** and then to **assess the treatment** in terms of *effort to deliver*, *complexity of treatment delivery* and *impacts to end users*. Treatment implementation tended to be structured to align with sprint goals and timelines and minimise negative or repeated impacts on end-users. To achieve **technical debt treatment work breakdown**, the team applyed *splitting strategies* and *grouping with enhancements*.

5. Discussion, Implications and Recommendations

We discuss our key findings about technical debt management in a data-intensive software team and triangulate these with with related works. We derive insights about how the team works and make some recommendations for practitioners and researchers that could improve the experience of such multidisciplinary teams and build relevant knowledge for the research community.

5.1. Mapping practitioner discussions about Technical Debt Types to research literature

Our study provides insight into the technical debt types experienced by a single data-intensive software team maintaining and extending a complex enterprise reporting product over a period of 6 weeks. We identified 5 sub-categories of technical debt types including: **technical data debt**, **data quality debt**, **pipeline debt**, **process debt** and **(non standard) documentation debt**. Based on the established taxonomies by (Rios et al., 2018; Li et al., 2015), our categories of **process debt**, **(non-standard) and documentation debt** can be readily aligned with their categories of Process Debt and Documentation Debt. We note that the *pipeline debt - sideloading* issues have been articulated at a high level by Scully et al. (Sculley et al., 2015), and recently explored by Foidl et al.'s review into pipeline quality and architectures (Foidl et al., 2024). Whilst data model debt type was articulated by Waltersdorfer (Waltersdorfer et al., 2020), their categorisation was limited to data model documentation and we note that further research is required to elaborate and generalise the categories of technical data debt.

➲ **Implications and Recommendation #1:** Much of the current technical data debt literature is based on data repository mining research. Whilst we are able to map some of our observed technical debt types to the technical debt types identified in existing literature, there are gaps regarding our findings pertaining to the **technical data debt** sub-category and *deployment pipeline* concept which also means that there is a lack of research backed guidance that can be translated or mapped to day to day work practices of data-intensive software teams.

We recommend that researchers further explore technical data debt concepts as experienced by practitioners to incorporate and map those perspectives to the findings from repository mining, so that practical solution anti-patterns, solution patterns and guidelines can be developed for practitioners. *We recommend* and urge practitioners to engage with and take an active part in these studies with researchers.

5.2. Use of TD and TD Management terminology

Our findings describe the observations that led us to conceptualise Technical Debt Types, and Technical Debt Management Categories including **Technical Debt Treatment Work**, **Identify and Assess Technical Debt** and **Technical Debt Treatment Work Breakdown**. Whilst basic technical debt terminology was used by team members, i.e. the use of phrases such as 'technical debt', 'refactor', and 'register of technical debt', we did not observe any use of advanced technical debt concepts e.g. discussions referring to financial metaphors such as 'repayment', 'interest', 'cost' or 'anti-patterns'. There were discussions about future refactoring efforts regarding when considering the potential to incur tech debt and ease of 'replumbing' components. In general, the work was discussed within the context of agile delivery and as such the language and focus are on the type of technical work they will do and the overall value it will deliver, such as 'cleaning up', 'removing legacy references', implementing new technology or processes and 'making consistent'. Our prior research identified that one of the characteristics of a multidisciplinary team is that maintaining communication between team members is typically very strong. Also, there is a lot of effort taken by team members to ensure understanding between team members that have different backgrounds (Graetsch et al., 2023). Whilst use of financial metaphors within the software engineering community may be accepted, our observations of a multidisciplinary DI team with several very experienced team members indicated that this has not become standard terminology in the data-intensive system space. Recently, Xavier et al. proposed and evaluated a lightweight framework to manage technical debt in agile software teams, where they incorporated an initial Technical Debt consensus step to allow the team to discuss and agree on technical debt definitions for their particular context (Xavier et al., 2023).

➲ **Implications and Recommendation #2:** The introduction of **technical debt focused language**, and associated financial metaphors could make it easier for team members to consider the longer term implications of taking on debt, or take advantage of downstream concepts such as anti-patterns to quickly recognise and assess technical debt and design-patterns that can

offer solutions. However, given the nature and varied backgrounds of multidisciplinary team members, care needs to be taken when introducing this terminology. *We recommend* that researchers and practitioners co-design technical debt concepts and vocabulary so that it can be adopted by multidisciplinary team members in a data-intensive environment. Given the agile delivery approach, the concepts and vocabulary need to be related to agile delivery and contain links to operational data work. A co-design based approach could be used to facilitate the development of this guidance.

5.3. Technical Debt identification and explicit documentation

We did not observe use of special tooling or techniques regarding technical debt identification in our case study. However, we did observe that knowledge about **known technical debt** was held in Jira tickets or attached documentation and significant tacit knowledge was evident during discussion of **anticipated technical debt**. Sometimes the discussions resulted in new backlog item creation, but not consistently. Team members in our study did express intent to document/collate technical debt i.e. make it explicit and the desire to hold broader discussions about technical debt i.e. with other teams. The team did not use a formal technical debt register (other than the backlog) and we did not observe any reference to agreed structures specifically for technical debt user stories.

The potential structure of technical debt documentation has been considered in software development focused literature such as Li et al. (Li et al., 2015) and the preference to keep a separate 'technical debt' backlog was discussed by Xavier et al. (Xavier et al., 2023). Whether and how well these approaches could meet the needs of data-intensive software has not been investigated. Authors of software development technical debt management frameworks do consider the creating and maintenance of a list of technical debt to be a foundational element of these frameworks (Guo et al., 2016; Xavier et al., 2023). However, the creation and maintenance have to be balanced against the effort of doing so (Guo et al., 2016).

➲ **Implications and Recommendation #3:** The need for a product manager to communicate about technical debt more broadly implies that information about identified technical debt needs to be shareable with broader audiences and technical debt management can have broader stakeholders than the immediate team. *We recommend* that practitioners consider the establishment of a technical debt register as part of their practices. Whilst further research is needed to clarify what information to capture for optimal downstream management and exactly what technical debt should be the focus - the need to have an explicit list of technical debt is foundational and it would reduce the risk of maintaining this knowledge tacitly. *We recommend* that the research community conduct further research studies to develop clear guidelines about what data intensive debt should be registered and what information should be captured.

Knowledge management strategies such as *documentation* and *knowledge sharing* sessions were also used by the team to address **(non standard) documentation debt**. Data-intensive software engineering is recognised as a highly knowledge intensive activity (Bruce and Mistrik, 2021) and there has been considerable research about the use of intelligent techniques to drive identification of software-intensive systems technical debt from artifacts, as synthesised by (Albuquerque et al., 2023). This research has not been extended to data-intensive systems and it has also not yet extended to converting tacit knowledge about technical debt to explicit knowledge.

➲ **Implications and Recommendation #4:** *We recommend* that researchers extend intelligent technical-debt identification studies to data-intensive debt and also consider research for converting tacit knowledge from discussions into explicit technical-debt documentation or registrations.

5.4. Multidisciplinary Data Development Tools support for TD Management

The team had modern development and collaboration tools (see Section 4.1.3), but **there was no observed support in the tools for identifying or managing technical debt**. As we did not observe individual team members perform their work individually, we did not observe whether the tools provided features to identify localised anti-patterns or remedy technical debt. However, we did observe team members perform *multidisciplinary review* sessions where they reviewed dashboards and SQL notebooks manually and identified naming inconsistencies, inconsistent configuration of widgets, and access to legacy components and we assumed that the tools either did not have the features or were not set up to guide team members to proactively flag errors.

Automated or context-sensitive tool-based support is highly dependent on research into data-intensive technical debt anti-patterns - which is still emerging. Muse et al. have identified SQL access anti-patterns through repository mining based research and planning to evaluate these findings with practitioners(Muse et al., 2022a).

➲ **Implications and Recommendation#5:** Data engineering tools have an important role in improving the recognition or warning about potential anti-patterns and inconsistencies and hence, prevention of technical debt. *We recommend* that researchers conduct further studies to articulate the appropriate, detailed configuration elements for organisational and team-based consistency rules to be incorporated into the tools and evaluate prototypes of tools. *We further recommend* that when developing the tools to identify technical debt, developers keep in mind the communication needs of multidisciplinary teams and their stakeholders so that any information produced is readily understood by a range of stakeholders. *We recommend* that tool developers enhance tools by incorporating features to allow configuration of organisational and team based data intensive consistency rules.

When team members were observed as they performed **multidisciplinary reviews**, one expert team member - usually the team member that had created or updated the SQL or PowerBI Dashboard, would 'drive' the tool in response to questions from other team members. There were no features within the tools to capture technical debt identification, enable collaboration, or incorporate other technical debt related feedback.

➲ **Implications and Recommendation #6:** The collaborative nature of review sessions where team members other than

the 'developer' reviewed aspects of SQL or visualisation configurations gives rise to the need for additional features in development tools that support these interactions. *We recommend* that researchers should evaluate whether collaborative review features such as ability to raise actions, questions and make comments that can be tracked by team members.

5.5. Technical Debt Treatment Work-Breakdown and Structuring

From a sprint delivery perspective, work items relating primarily to technical debt treatment in our case study were assigned to and measured as 'continuous improvement work'. For the prior 4 months, continuous improvement work was kept at 8% to 9% of sprint capacity, which equates to roughly 1 or 2 Jira tickets per sprint, depending on the story points assigned.

The identified technical debt treatment work which had been captured in stories by individual team members was considered and scrutinised by different team members during Backlog Refinement and Sprint Planning ceremonies. The multidisciplinary perspectives of the team members contributed to different considerations and led to the identification of how components inter-relate and impact. These discussions identified that further refinement, *splitting*, or *grouping* were required to fit within sprint capacity, risk parameters and value delivery parameters. In effect, the team structured the work to fit into the parameters of the sprint. Whilst the action of splitting and restructuring the debt can be related to prioritisation of technical debt, which has been studied both in the context of software-engineering technical debt management (Besker et al., 2019) and data-related technical debt (Albarak and Bahsoon, 2018), prioritisation concepts and regimes only consider how to prioritise identified technical-debt. They do not address how to split and how to structure work, and are hence insufficient to support these activities.

➲ **Implications and Recommendation #7:** *We recommend* further research to develop approaches and patterns to break down or structure technical debt treatment work, in effect, to develop technical debt repayment strategies for complex data-intensive system evolution. This could include staged implementations, or 'refinance strategies' that replace highly risky technical debt, with lower grade technical debt. The aim is to provide options that can be accommodated based on meeting sprint capacity. The strategies should also cater for the multidisciplinary nature of DI teams and allow staging and targeting to available expertise.

6. Threats to Validity and Limitations

We discuss the major threats to the validity of our findings arising out of both the Case Study approach, and our use of the Socio-Technical Grounded Theory method for our data analysis.

6.1. Case Study

We present key threats to validity for our study based on the classification scheme proposed by (Yin, 2018), as recommended by Runeson et al. (Runeson and Höst, 2009):

Construct validity in the context of a case study refers to whether the case study identified the right operational measures and right concepts being studied (Yin, 2018; Runeson and Höst, 2009). Our case study's overarching goal was to study the phenomena of a multidisciplinary team working together and dealing with challenging data scenarios when delivering data-intensive software. Given that the team used the Agile delivery, attendance at ceremonies offered the appropriate starting point for observations, followed by deep dives. During the inductive analysis, our attention focused on building concepts about technical debt. We utilised a number of data sources - recorded observations (including Jira card extracts, documentation extracts and tool screenshots), clarification discussions and team Slack messages. We also made materials available to team members at the conclusion of the observations, and provided a draft report with findings to participants and offered presentations of our findings. Five participants attended these presentations.

Internal validity in the context of a case studies refers to causal relations examined and whether any factors not considered affect the investigated factor (Yin, 2018; Runeson and Höst, 2009). Our observations spanned many team interactions during a 6 week period, but not all. There were a number of meetings held that the team or team members attended outside their dedicated Agile ceremonies and as such we need to recognise that our observations were incomplete and could have missed factors that shaped the discussions but were not overtly discussed.

Reliability is concerned with to what extent the data and the analysis are dependent on the specific researchers and repeatability of our (Yin, 2018; Runeson and Höst, 2009). Our results are supported through the design, documentation and application of our case study protocol and associated data management process. Whilst the data collection was carried out by a single researcher, it followed protocol and the protocol documentation was reviewed by the project team and has been made available in supplementary materials and could support the repeatability of the case study in a different context. As the case study was conducted within a 'live' environment, an identical study cannot be repeated. Further, due to privacy and commercial confidentiality, we are unable to make the data set available for repeat analysis.

External validity is concerned with to what extent it is possible to generalise the findings (Yin, 2018; Runeson and Höst, 2009). Our single case study cannot make claims about generalisation of results. Instead, this case study has aimed to explore details of multidisciplinary team practices and build knowledge to support contextualising work and we have provided significant contextual details to meet this aim. The case study was motivated by our theory about multidisciplinary teams dealing with data challenges (Graetsch et al., 2023) and structured to better understand the context and domain of work practices of multidisciplinary teams developing data-intensive software solutions.

6.2. Socio-Technical Grounded Theory Method

The application of STGT method should be evaluated through the demonstrating *credibility* and *rigour* (Hoda, 2022). We provide evidence of *rigour* by providing coding examples and sanitised quotes throughout our findings. We demonstrate *credibility* of method application by articulating how we collected and filtered the data, and interleaved coding and memoing to derive concepts.

Using a limited application of STGT for data analysis within an observational case study, our findings represent an emerging theoretical model. Guidelines for evaluating emerging findings include the criteria of *originality*, *relevance* and *density*. Our findings demonstrate *originality* because we conducted an original research study and collected first hand data of a multidisciplinary team discussing their data intensive software development work. Our study is placed in an original context of a multidisciplinary DI software team and offers an original perspective on DI technical debt and its management. Similar to demonstrating the construct validity of the case study above, one way to demonstrate *relevance* of the outcome of the analysis is to receive feedback from participants that validate findings (Hoda, 2022). We made materials available to team members at conclusion of the observations, by providing a draft technical report with findings to the project team members and organisation contact and sought their feedback. The final evaluation criteria, *density* is achieved when a category is supported by multiple concepts and properties that capture a range of contexts and nuances. The descriptions of the categories should include pertinent examples and evidence from underlying data (Hoda, 2022). Our findings meet this criteria.

6.3. Other Limitations

Even though the participants were assured that they would be anonymised, there is a threat that participants did not speak freely during the observations or clarifying interviews. Also, even though the participants are anonymised for readers who did not partake in the study, they are still likely to be able to identify each other and attribute quotes to each other from the materials presented in the case study. As part of the informed consent process, each participant was made aware that they would be observed and that observation sessions would be recorded. Prior to each observation session, the first author made participants aware that the session would be recorded - hence ensuring that participants had awareness of the recording.

Further, the first author, who has extensive experience in project delivery, conducted the observations and performed all coding. This poses the risk of bias. The choice of STGT method mitigates this risk because it allows for and expects researchers to have some expertise to enable filtering of data and conceptualisation. Further, the risk was mitigated through extensive discussions and reviews of codes and concepts with the other authors.

7. Conclusion

We designed and conducted a 6-week observational case study of a 12 team member multidisciplinary team that was developing a data analytics system. We present detailed contextual descriptions of the case context including organisational structures, product and stakeholder information, technology architecture and tools, team member backgrounds, as well as a summary of how the team works in terms of its ceremonies, the work they perform and how the team's work is measured from an organisational perspective. We applied the Socio-Technical Grounded Theory Method for data analysis to develop categories and concepts to categorise types of observed technical debt and technical debt management. Our findings provide a novel perspective on how a multidisciplinary Data-Intensive software team manages technical debt.

We identified technical debt types that the team deals with, including technical data debt, pipeline debt, process debt, data quality debt and (non) standard document debt. We conceptualised how the team manages technical debt, including through identification and assessment of technical debt, technical debt work, and technical debt treatment work breakdown. By taking this approach, we conceptualised that the team assesses known, anticipated and unanticipated debt differently. We identified technical debt data treatments and related treatment quality assurance activities, which are part of technical debt treatment work, a broader category than traditional debt repayment. Technical debt treatments which includes techniques to avoid technical debt, knowledge management strategies and environment enhancements. Technical debt treatments work evolves and can be influenced by factors outside the team's immediate control, such as organisational communities of practice and technology vendors. We also identified insights into how multidisciplinary perspectives of the team members contribute to grouping and splitting of technical debt treatment so that it can fit within sprint parameters.

Our findings and consideration of the literature raise important implications for the technical debt language used by teams, how technical debt should be documented, tool requirements and support requirements for structuring technical debt treatment.

Acknowledgements

We acknowledge and thank the organisation executive leadership and especially the participant team members for being open and generous with access to their working lives. We also thank the stakeholders who participated in ceremonies for granting us permission to conduct our observation study.

The first author would like to acknowledge Prof. Yvonne Dittrich for the time taken to discuss and share insights on practice theory, conducting observations and data analysis.

Graetsch was supported by a Faculty of IT PhD scholarship. Grundy is supported by ARC Laureate Fellowship FL190100035.

References

Albarak, M., Bahsoon, R., 2016. Database design debts through examining schema evolution, in: 2016 IEEE 8th International Workshop on Managing Technical Debt (MTD), pp. 17–23. doi:10.1109/MTD.2016.9.

Albarak, M., Bahsoon, R., 2018. Prioritizing technical debt in database normalization using portfolio theory and data quality metrics, in: Proceedings of the 2018 International Conference on Technical Debt, ACM, New York, NY, USA. pp. 31–40. doi:https://doi.org/10.1145/3194164.3194170.

Albarak, M., Bahsoon, R., Ozkaya, I., Nord, R.L., 2020. Managing technical debt in database normalization. IEEE Trans. Softw. Eng. 48, 1–1. doi:10.1109/TSE.2020.3001339.

Albuquerque, D., Guimarães, E., Tonin, G., Rodríguezs, P., Perkusich, M., Almeida, H., Perkusich, A., Chagas, F., 2023. Managing technical debt using intelligent techniques - a systematic mapping study. IEEE Trans. Softw. Eng. 49, 2202–2220.

Alves, N.S.R., Mendes, T.S., de Mendonça, M.G., Spínola, R.O., Shull, F., Seaman, C., 2016. Identification and management of technical debt: A systematic mapping study. Inf. Softw. Technol. 70, 100–121. doi:https://doi.org/10.1016/j.infsof.2015.10.008.

Aniche, M.F., Oliva, G.A., Gerosa, M.A., 2014. Are the methods in your data access objects (DAOs) in the right place? a preliminary study, in: 2014 Sixth International Workshop on Managing Technical Debt, IEEE. pp. 47–50. doi:10.1109/MTD.2014.14.

Avgeriou, P., Kruchten, P., Ozkaya, I., Seaman, C., 2016. Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162). Dagstuhl Reports 6, 110–138. doi:https://doi.org/10.4230/DagRep.6.4.110.

Besker, T., Martini, A., Bosch, J., 2019. Technical debt triage in backlog management, in: 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), IEEE. pp. 13–22. doi:https://doi.org/10.1109/TechDebt.2019.00010.

Bruce, R., Mistrik, B., 2021. Knowledge management in the development of data-intensive systems. 1st edition ed., Auerbach, London, England.

Charmaz, K., 2014. Constructing grounded theory. Introducing qualitative methods. 2nd edition ed., Sage.

Choi, B., Pak, A., 2006. Multidisciplinarity, interdisciplinarity and transdisciplinarity in health research, services, education and policy: 1. definitions, objectives, and evidence of effectiveness. Clin Invest Med 29, 351–364.

Corbin, J.M., 2008. Basics of qualitative research : techniques and procedures for developing grounded theory. 3e [ed.] / juliet corbin, anselm strauss. ed.

Cunningham, W., 1992. The wycash portfolio management system. SIGPLAN OOPS Mess. 4, 29–30. doi:https://doi.org/10.1145/157710.157715.

Dittrich, Y., Michelsen, C.B., Tell, P., Lous, P., Ebdrup, A., 2020. Exploring the evolution of software practices, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Association for Computing Machinery, New York, NY, USA. p. 493–504. doi:https://doi.org/10.1145/3368089.3409766.

Fetterman, D.M., 2019. Ethnography: Step-by-step. Sage publications.

Foidl, H., Felderer, M., Biffl, S., 2019. Technical debt in data-intensive software systems, in: 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 338–341. doi:10.1109/SEAA.2019.00058.

Foidl, H., Golendukhina, V., Ramler, R., Felderer, M., 2024. Data pipeline quality: Influencing factors, root causes of data-related issues, and processing problem areas for developers. J. Syst. Softw. 207, 111855.

Fortune, 2023, 2023. Technology big data analytics market - market research report summary. URL: `https://www.fortunebusinessinsights.com/big-data-analytics-market-106179`. accessed on: 2024-07-28.

Freire, S., Rios, N., Pérez, B., Castellanos, C., Correal, D., Ramač, R., Mandić, V., Taušan, N., López, G., Pacheco, A., Mendonça, M., Falessi, D., Izurieta, C., Seaman, C., Spínola, R., 2023. Software practitioners' point of view on technical debt payment. Journal of Systems and Software 196, 111554. doi:https://doi.org/10.1016/j.jss.2022.111554.

Glaser, B., Strauss, A.L., 2017. The discovery of grounded theory : strategies for qualitative research. Sociology Press.

Graetsch, U.M., Khalajzadeh, H., Shahin, M., Hoda, R., Grundy, J., 2023. Dealing with data challenges when delivering data-intensive software solutions. IEEE Transactions on Software Engineering 49, 4349–4370. doi:10.1109/TSE.2023.3291003.

Guo, Y., Spínola, R.O., Seaman, C., 2016. Exploring the costs of technical debt management – a case study. Empir. Softw. Eng. 21, 159–182. doi:https://doi.org/10.1007/s10664-014-9351-7.

Gupta, A., 2022. Why fusion teams matter. Gartner. URL: `https://www.gartner.com/en/articles/why-fusion-teams-matter`. access: 2024-07-28.

Hammersley, M., 2020. Ethics of ethnography, in: Handbook of Research Ethics and Scientific Integrity. Springer International Publishing, Cham, pp. 445–457.

Hoda, R., 2022. Socio-technical grounded theory for software engineering. IEEE Transactions on Software Engineering 48, 3808–3832. doi:https://doi.org/10.1109/TSE.2021.3106280.

Hoda, R., 2024. Qualitative Research with Socio-Technical Grounded Theory - A Practical Guide to Qualitative Data Analysis and Theory Development in the Digital World. Springer (Releasing 2024).

Hutchison, A.J., Johnston, L.H., Breckon, J.D., 2010. Using qsr-nvivo to facilitate the development of a grounded theory project: an account of a worked example. International journal of social research methodology 13, 283–302.

Kenyon, T., 2021. Big data and business analytics to hit US $275B in 2022. AI Magazine. URL: `https://aimagazine.com/data-and-analytics/big-data-and-business-analytics-hit-usdollar274bn-2022`. accessed on: 2024-07-28.

Kimball, R., Margy, R., Warren, T., Joy, M., Bob, B., 2008. The data warehouse lifecycle toolkit. Second edition. ed., Wiley., Indianapolis, Ind.

Kleinwaks, H., Batchelor, A., Bradley, T.H., 2023. Technical debt in systems engineering—a systematic literature review. Syst. Eng. 26, 675–687. doi:https://doi.org/10.1002/sys.21681.

Li, Z., Avgeriou, P., Liang, P., 2015. A systematic mapping study on technical debt and its management. J. Syst. Softw. 101, 193–220. doi:https://doi.org/10.1016/j.jss.2014.12.027.

Muse, B.A., Nafi, K.W., Khomh, F., Antoniol, G., 2022a. Data-access performance anti-patterns in data-intensive systems. arXiv [cs.SE] doi:https://doi.org/10.48550/arXiv.2208.08918.

Muse, B.A., Nagy, C., Cleve, A., Khomh, F., Antoniol, G., 2022b. FIXME: synchronize with database! an empirical study of data access self-admitted technical debt. Empir. Softw. Eng. 27, 1–42. doi:https://doi.org/10.1007/s10664-022-10119-4.

Nippert-Eng, C., 2015. Watching closely: A guide to ethnographic observation. Oxford University Press.

Recupito, G., Pecorelli, F., Catolino, G., Lenarduzzi, V., Taibi, D., Di Nucci, D., Palomba, F., 2024. Technical debt in ai-enabled systems: On the prevalence, severity, impact, and management strategies for code and architecture. Journal of Systems and Software 216, 112151. doi:https://doi.org/10.1016/j.jss.2024.112151.

Rios, N., Mendonça Neto, M.G.d., Spínola, R.O., 2018. A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. Inf. Softw. Technol. 102, 117–145. doi:https://doi.org/10.1016/j.infsof.2018.05.010.

Runeson, P., Host, M., Rainer, A., Regnell, B., 2012. Case Study Research in Software Engineering: Guidelines and Examples. 1. ed., Wiley, Newark.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14, 131–164. doi:https://doi.org/10.1007/s10664-008-9102-8.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.F., Dennison, D., 2015. Hidden technical debt in machine learning systems. Neural Inf Process Syst , 2503–2511.

Seaman, C., Guo, Y., 2011. Chapter 2 - measuring and monitoring technical debt, in: Zelkowitz, M.V. (Ed.), Advances in Computers. Elsevier. volume 82 of *Advances in Computers*, pp. 25–46. doi:https://doi.org/10.1016/B978-0-12-385512-1.00002-5.

Shah, H., Harrold, M.J., Sinha, S., 2014. Global software testing under deadline pressure: Vendor-side experiences. Information and Software Technology 56, 6–19. doi:https://doi.org/10.1016/j.infsof.2013.04.005.

Sharp, H., Dittrich, Y., de Souza, C.R.B., 2016. The role of ethnographic studies in empirical software engineering. IEEE Transactions on Software Engineering 42, 786–804. doi:https://doi.org/10.1109/TSE.2016.2519887.

Sharp, H., Robinson, H., 2004. An ethnographic study of xp practice. Empirical Software Engineering 9, 353–375. doi:https://doi.org/10.1023/B:EMSE.0000039884.79385.54.

Sklavenitis, D., Kalles, D., 2024. Measuring technical debt in ai-based competition platforms. URL: `https://arxiv.org/abs/2405.11825`,

arXiv:2405.11825.
Soliman, J., Kan, M., 2004. Grounded theory and nvivo: wars and wins, in: International conference on qualitative research in IT & IT in Qualitative Research, pp. 24–26.
Spradley, J.P., 2016. Participant observation. Waveland Press.
Storey, M.A., Ernst, N.A., Williams, C., Kalliamvakou, E., 2020. The who, what, how of software engineering research: a socio-technical framework. Empir. Softw. Eng. 25, 4097–4129. doi:https://doi.org/10.1007/s10664-020-09858-z.
Tang, Y., Khatchadourian, R., Bagherzadeh, M., Singh, R., Stewart, A., Raja, A., 2021. An empirical study of refactorings and technical debt in machine learning systems, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 238–250. doi:https://doi.org/10.1109/icse43902.2021.00033.
Tom, E., Aurum, A., Vidgen, R., 2013. An exploration of technical debt. J. Syst. Softw. 86, 1498–1516. doi:https://doi.org/10.1016/j.jss.2012.12.052.
Waltersdorfer, L., Rinker, F., Kathrein, L., Biffl, S., 2020. Experiences with technical debt and management strategies in production systems engineering, in: Proceedings of the 3rd International Conference on Technical Debt, Association for Computing Machinery, New York, NY, USA. p. 41–50. doi:https://doi.org/10.1145/3387906.3388627.
Weber, J.H., Cleve, A., Meurice, L., Bermudez Ruiz, F.J., 2014. Managing technical debt in database schemas of critical software, in: 2014 Sixth International Workshop on Managing Technical Debt, IEEE. pp. 43–46. doi:https://doi.org/10.1109/MTD.2014.17.
Wohlin, C., 2021. Case study research in software Engineering—It is a case, and it is a study, but is it a case study? Inf. Softw. Technol. 133, 106514. doi:https://doi.org/10.1016/j.infsof.2021.106514.
Xavier, L., dos Santos, R., Bessa, S., Tulio Valente, M., 2023. Agile technical debt management using the LTD framework. Softw. Eng. Notes 49, 13–23. doi:https://doi.org/10.1145/3635439.3635443.
Yin, R.K., 2018. Case study research and applications : design and methods. Sixth edition. ed., SAGE Publications, Inc., Thousand Oaks, California.

Ulrike M. Graetsch is a PhD student at Monash University, Faculty of IT, Humanise Lab. She completed her undergraduate and Masters of Computing at Monash University. Since graduating, she gained over 20 years of experience as a practitioner in IT Project Delivery. She has a keen interest in data analytics software delivery and ensuring that multidisciplinary teams can work effectively to develop solutions that meet the needs of diverse end-users.

Rashina Hoda is a Professor in Software Engineering at Monash University, Australia. Rashina specialises in human-centered empirical software engineering and has introduced "Socio-Technical Grounded Theory" (STGT), as a modern variant of the traditional sociological GT methods. She received an ACM SIGSOFT Distinguished Paper Award (ICSE 2017) and Distinguished Reviewer Award (ICSE 2020). She serves as an Associate Editor of the IEEE Transactions on Software Engineering and the Workshops co-chair at ICSE 2024. Previously, she served on the IEEE Software Advisory Board, as Associate Editor of Journal of Systems and Software, SEIS PC co-chair at ICSE 2023, CHASE 2021 PC co–chair and XP2020 PC co–chair. More details on https://rashina.com.

Hourieh Khalajzadeh is a Senior Lecturer in the School of Information Technology at Deakin University. Previously, she was a Research Fellow in the HumaniSE Lab at Monash University. Hourieh's research is situated at the intersection of software engineering and data science. She is currently looking at the human-centric issues in Software Engineering and is experienced in designing domain specific visual languages for different applications, including big data analytics development.

Mojtaba Shahin is a Lecturer in the School of Computing Technologies at RMIT University, Melbourne. Previously, he was a Research Fellow at Monash University. His research interests reside in Empirical Software Engineering, Human and Social Aspects of Software Engineering, and Secure Software Engineering. He completed his PhD study at the University of Adelaide, Australia.

John Grundy is Australian Laureate Fellow and Professor of Software Engineering at Monash University. He leads the HumaniSE lab in the Faculty of Information Technology, investigating "human-centric" issues in software engineering. These include impact of personality on software engineers and users; emotion-oriented requirements engineering; impact of different languages, cultures and belief sets on using and engineering software; usability and accessibility of software, particularly for ageing people and people with physical and mental challenges; issues of gender, age, socio-economic status and personal values on software, software requirements, and software engineering teams.