

Combining Requirements Engineering Techniques for the Analysis of a Legacy System

Jessica Friedline, Jan-Philipp Steghöfer*

XITASO GmbH, Austraße 35, 86153 Augsburg, Germany

Abstract

[Context and Motivation] Software consultancies often analyze undocumented legacy systems that have evolved over many years. **[Question/problem]** The lack of documented requirements, processes, and design decisions make reverse engineering these systems for analysis or reimagination a complex task. **[Principal ideas/result]** We detail how we utilized various requirements engineering techniques, including the development of domain stories from system screencasts, user story mapping, and process descriptions enhanced through interviews and contextual inquiries. We followed an iterative approach, with regular stakeholder feedback to fine-tune our outputs. **[Contribution]** We demonstrate the effectiveness of integrating these methods in legacy system analysis, sharing insights and key takeaways derived from our approach.

Keywords

Legacy systems, Re-engineering, Domain Storytelling, User Story Mapping, Contextual Inquiry

1. Introduction

Legacy systems encapsulate critical business processes, making their replacement a complex task [1]. Requirements engineering plays a crucial role in documenting these systems for transformations or to maintain existing functionalities [2]. There is some related work on extracting requirements from legacy systems: some studies explore reverse engineering [3, 4], one study uses observations of the running system and user interactions [5], and another one uses process mining based on source code analysis [6]. However, these papers provide little practical guidance that can be applied across a range of scenarios.


In this report, we detail our methodology for analysing a legacy system. Combining varied data collection and analysis techniques, the methodology caters to diverse stakeholder needs. For data collection we employed document analysis, screencasts, interviews, and field observations (see Section 4). To analyze this data we utilized domain storytelling [7], user story mapping [8], and process modeling (see Section 5). We describe how we combined these techniques to create a full picture of the legacy system that is accessible to technical and non-technical stakeholders

In: D. Mendez, A. Moreira, J. Horkoff, T. Weyer, M. Daneva, M. Unterkalmsteiner, S. Bühne, J. Hehn, B. Penzenstadler, N. Condori-Fernández, O. Dieste, R. Guizzardi, K. M. Habibullah, A. Perini, A. Susi, S. Abualhaija, C. Arora, D. Dell'Anna, A. Ferrari, S. Ghanavati, F. Dalpiaz, J. Steghöfer, A. Rachmann, J. Gulden, A. Müller, M. Beck, D. Birkmeier, A. Herrmann, P. Mennig, K. Schneider. Joint Proceedings of REFSQ-2024 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. Co-located with REFSQ 2024. Winterthur, Switzerland, April 8, 2024.

*Corresponding author.

✉ jessica.friedline@xitaso.com (J. Friedline); jan-philipp.steghoefer@xitaso.com (J. Steghöfer)

ORCID [0000-0003-1694-0972](https://orcid.org/0000-0003-1694-0972) (J. Steghöfer)

 © 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

and which lessons we learned during the project (see Section 6). Our approach most closely resembles Liu et al. [5] as it relies on observation and modelling, but uses different data collection and analysis methods and is arguably more pragmatic.

2. Case Description

Our client contracted us to document a decade-old legacy system crucial for constructing and validating machinery. The system is actively used by several hundred employees across various departments and involves significant investments. It is built on top of a database management system and uses the built-in user interface features to provide a series of forms for data entry and review. Despite functioning adequately, the system's sparse documentation and lack of IT department oversight posed risks, particularly because it supports critical business processes and serves as an information hub.

The client requested that we produce a number of documents to serve as the foundation for a decision about the system's future:

- **Software architecture and design documentation** detailing the structure of the system, its components, interfaces to other systems, and implementation choices.
- **Requirements specifications** that capture the most important high-level requirements the system currently fulfills.
- **Process descriptions** that detail the important business processes the system supports.
- **Future needs** that describe which features and changes need to be implemented to address upcoming requirements and technical needs.

For this consultancy commission, our team consisted of two analysts and a project owner, with the latter primarily focusing on project management tasks. To initiate the analysis, the client supplied us with roughly 100 documents comprising system presentations, data sharing agreements, a developer-answered IT questionnaire, prior process analysis attempts, and a screencast of a system demo given by one of its developers. Our client also gave us access to the developers and some users of the system. However, due to security concerns, our client did not share the source code, limiting our ability to create detailed technical documentation and utilize automated analysis methods [6, 4].

3. Iterative Approach

We followed an iterative-incremental approach from the outset: During an initial kick-off meeting with the system developers, their manager, and IT representatives, we outlined our understanding of the tasks and our proposed workflow. Their feedback and the discussions allowed us to refine our approach, particularly regarding the documents we needed to produce. An overview of our approach is illustrated in Figure 1.

The client emphasized the need for regular updates. An online *Kanban board* was set up to track the progress, with epics for architecture, design, requirements, process, and future needs, and stories for each of the requested documents. *Wiki pages* were created for each deliverable and regularly updated to keep the customer informed of our ongoing analysis. We scheduled

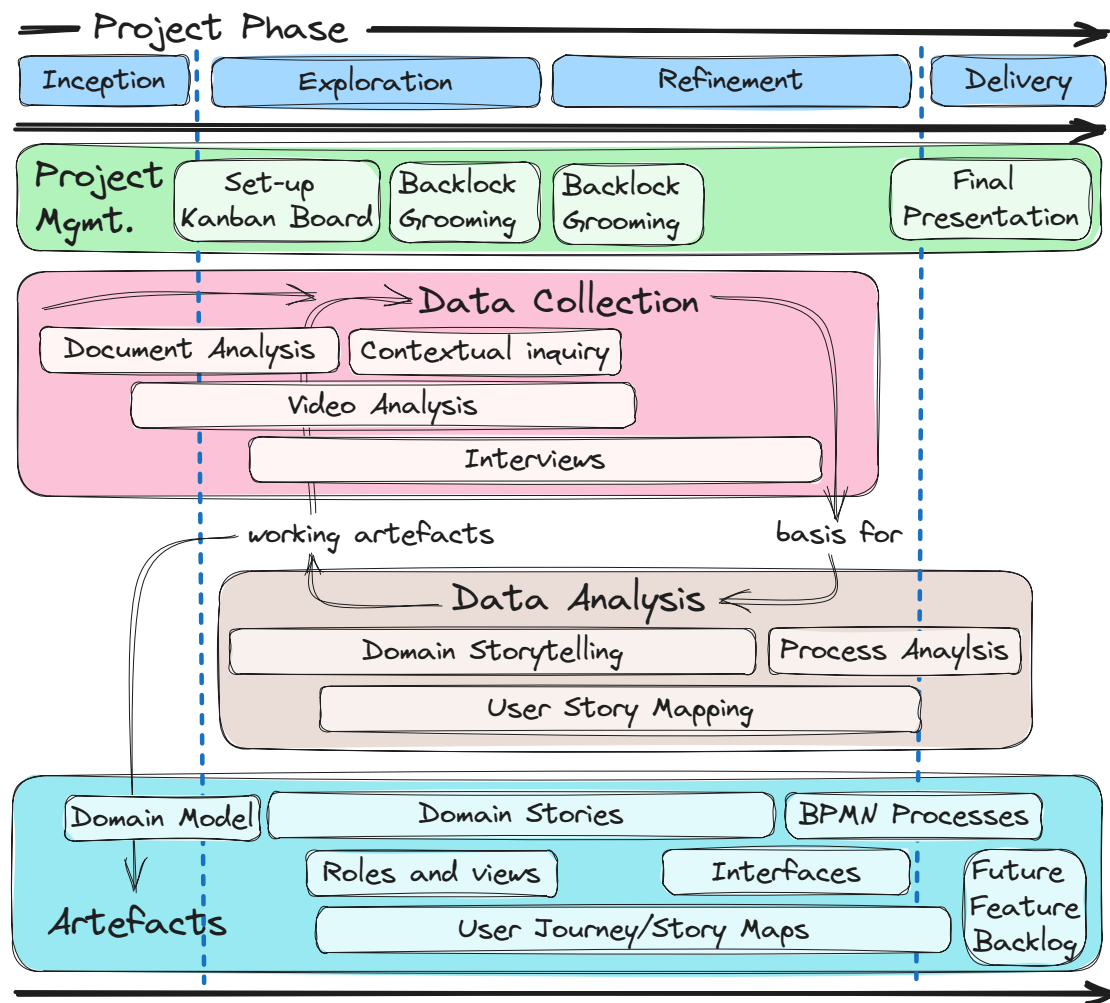


Figure 1: Project phases, activities, and artifacts, indicating the focus and creation times for each. Artifacts were developed incrementally, and activities were iterative.

regular *semi-structured interviews* with the two developers, using their feedback for refinement. We also conducted an *on-site visit* for contextual inquiry where we were able to validate the most important business processes and observe how end-users of the system interacted with it. Backlog grooming meetings with IT representatives helped refine the scope, document the decisions and further clarified the scope of the final delivery. The project concluded with a final presentation of our results, where we received our sign-off.

4. Data Collection Methods

4.1. Document Analysis

We initiated our document analysis by evaluating around 100 documents supplied by the client and categorizing them by utility. We found that around 38% were email archives discussing project management. Around 20% were Microsoft Word documents, about half of which were useful, because they contained data sharing agreements or meeting summaries, while the rest was mostly concerned with project management. Microsoft PowerPoint presentations made up 18% of the documents, which included both project management content (e.g., staffing) and some relevant information, e.g., about the overall IT architecture. 12% were PDF files, largely duplicative or unrelated. Excel spreadsheets accounted for 7% and contained partial results of a previous analysis attempt and were disregarded on request of the customer.

The number of useful documents came to 25 which two analysts could analyse in a reasonably short time. Their content was mostly unstructured so that typical automated approaches, e.g., using natural language processing (see, e.g., [9]) were not useful. Instead, we manually extracted the information that was deemed relevant in short summaries to a shared document.

4.2. Video analysis

The standout document was a 90-minute screencast which offered comprehensive insights into the system. The developer systematically demonstrated most of the system's relevant functionality and explicitly discussed the differences in how the unique roles interacted with the system. The developer used domain terms to discuss many aspects and the discussion did not dive deeply into implementation details, with the exception of a discussion how the system retrieves information from the company's LDAP server.

The demonstration made it possible for us to start systematizing the information with a focus on the involved roles, the role-specific views, and processes the system supports. We captured this information in domain story diagrams (see Section 5.1), highlighting role-specific system access which informed our initial business process drafts. We started sketching a domain model to capture the vocabulary and the relationships between the concepts of the domain. Although incomplete, this model was useful in the exploratory phase of our research.

4.3. Interviews

Productive semi-structured interviews with the developers became the main driver of our iterative-incremental production of artifacts. Document and video analysis allowed us to quickly produce draft versions of the required artifacts, in particular of the domain stories. These drafts were then validated and refined based on the interviews, usually when we had completed another chunk of analysis we wanted to discuss. In many cases, we had specific questions that we wanted to discuss. Live editing of these drafts during the interviews proved to be particularly helpful: The interviewees could directly comment on the changes one analyst made while the other analyst conducted the interview.

4.4. Contextual Inquiry

Through Contextual inquiry we were able to witness users in their normal work environments, revealing crucial functional details about how the system is used to complete the different workflows of the users [10]. In particular, we were able to we enhance and validate the domain stories and user story maps during our visit. Across departments and in the spaces where complex machinery is assembled and tested, we were able to watch some of the power users of the system interact with it to complete their tasks, which also allowed us to discover the tools which are used to complement the software and bridge gaps within its functionality: Some communication vital for the processes was in part documented elsewhere, for example, including the availability of test beds for the assembled machines, which was mostly organized by phone and documented by one department head in a custom spread sheet which supported this task with a number of macros.

5. Data Analysis Methods

5.1. Domain Story Telling

Domain storytelling [7] played a significant role in our reasearch: From the perspective of the business domain, key aspects of a system are described in a structured and visual way, using easy to understand graphs and narrative descriptions of processes, roles/actors, and their interactions with different subdomains. We mapped out the various functions and screens of the software as areas with which the different users/roles interacted and described their interactions with narrative labels on graphs, which point from the actor to the area of interest and are also numbered to show the sequence. The domain stories resonated with our customer and the interviewees because they are easy to understand, written in natural language, and use the domain appropriate wording. This enabled us to efficiently verify our understanding of the different business processes and how the software supports them.

5.2. User Story Mapping

User story mapping [8], enriched with aspects of user journey mapping [11], helped construct a narrative of user experiences, external and internal points of software interaction, and the sequential flow crucial for comprehensive system understanding. This greatly helped in preparing for our user interviews and contextual inquiries, and in our understanding of the software. The final artefact also included screenshots from the software and thus provided our customer with an in-depth understanding of the look and feel of the software and the steps needed to fulfill the most important tasks.

5.3. Process Modelling

We represented business processes using BPMN, drawing on our domain stories as a source. BPMN diagrams make it easier to model case distinctions than domain stories and we made extensive use of BPMN messaging events to model the communication pathways between the roles. However, they proved less intuitive for stakeholder feedback compared to domain stories.

6. Discussion

During our experience, we made a number of observations which are supported by existing literature. Among them are that management-oriented documents often lack the granular information required for technical analysis [12] and that an iterative-incremental approach with easy-to-understand artifacts enhances efficiency and transparency, building stakeholder trust. The latter is a common tenet in agile software engineering. However, we also found aspects that are not reported in literature and present unique lessons learned about requirements analysis for legacy systems and the usefulness of artifact redundancy in client communication and information capture.

Screencasts as a source for requirements: The screencast proved to be a valuable tool in reconstructing requirements, complementing the existing knowledge on how videos enhance requirements engineering practices [13, 14, 15].

Observation: To the best of our knowledge, the current body of work does not address using analysis of system-screencasts to reconstruct requirements and domain information.

Lesson Learned: The screencast proved valuable in all phases of our analysis, enabling us to revisit specific topics to discover details and answer questions that arose with a deeper understanding of the system over time.

Contextual inquiry as a powerful tool: Contextual inquiry provided a full view of user workflows and software limitations, proving critical for understanding complex systems where interviews may miss nuanced workflow elements [8].

Observation: We discovered that gaps in our understanding often aligned with corresponding gaps in the software’s functionality.

Lesson Learned: Contextual inquiry is essential in legacy software analysis to comprehend entire workflows and identify software functionality gaps.

Redundancies in produced artifacts: We were initially reluctant to add artifacts that capture similar information. In particular, the client’s request to create BPMN diagrams to depict the business processes seemed to replicate the domain stories we had already created. In general, redundant information is frowned upon in software engineering [16] since it introduces issues with maintainability and consistency. However, in the analysis of legacy systems, the need to keep artifacts consistent over time is less pronounced.

Observation: Artifact redundancy was beneficial. Domain stories, BPMN diagrams, and user story maps each provided unique insights despite overlapping information, catering to different stakeholder needs and purposes.

Lesson Learned: Selecting purpose-specific artifacts adds value to both the analysis process and stakeholder understanding for legacy systems.

7. Conclusion

In this short paper, we discuss our preliminary results of combining different requirements engineering techniques, including interviews, contextual inquiry, domain storytelling, and user story mapping, in the analysis of a legacy software system. Our lessons learned include the benefits of screencasts of the system in action, understanding when in the analysis contextual inquiry is valuable, and realizing the utility of redundant artifacts in analyzing legacy systems. We believe our experiences will be useful to practitioners that are faced with a similar tasks and researchers interested in sharpening the tools at the disposal of requirements engineers in the field. As future work, we are going to extend our analysis and combine our preliminary results with lessons learned from a second case in which we employed the same iterative approach.

References

- [1] S. Matthiesen, P. Bjørn, Why replacing legacy systems is so hard in global software development: An information infrastructure perspective, in: 18th ACM Conf. on Computer-supported Cooperative Work & Social Computing, 2015, pp. 876–890.
- [2] A. Alexandrova, L. Rapanotti, Requirements analysis gamification in legacy system replacement projects, *Requirements Engineering* 25 (2020) 131–151.
- [3] S. A. Fahmi, H.-J. Choi, Software reverse engineering to requirements, in: *Int. Conf. on Convergence Information Technology (ICCIT 2007)*, IEEE, 2007, pp. 2199–2204.
- [4] S. Hassan, U. Qamar, T. Hassan, M. Waqas, Software reverse engineering to requirement engineering for evolution of legacy system, in: 2015 5th Int. Conf. on IT Convergence and Security (ICITCS), IEEE, 2015, pp. 1–4.
- [5] K. Liu, A. Alderson, Z. Qureshi, Requirements recovery from legacy systems by analysing and modelling behaviour, in: *IEEE Int. Conf. on Software Maintenance (ICSM'99)*, IEEE, 1999, pp. 3–12.
- [6] A. C. Kalsing, G. S. do Nascimento, C. Iochpe, L. H. Thom, An incremental process mining approach to extract knowledge from legacy systems, in: 14th IEEE Int. Enterprise Distributed Object Computing Conference, IEEE, 2010, pp. 79–88.
- [7] S. Hofer, H. Schwentner, *Domain Storytelling: A Collaborative, Visual, and Agile Way to Build Domain-Driven Software*, Pearson Education, Limited, 2021.
- [8] A. Milicic, A. Perdikakis, S. E. Kadiri, D. Kiritsis, P. Ivanov, Towards the definition of domain concepts and knowledge through the application of the user story mapping method, in: *IFIP Int. Conf. on Product Lifecycle Management*, Springer, 2012, pp. 58–69.
- [9] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, R. T. Batista-Navarro, Natural language processing for requirements engineering: A systematic mapping study, *ACM Comput. Surv.* 54 (2021).
- [10] P. M. Bednar, C. Welch, Contextual inquiry and requirements shaping, in: *Information Systems Development*, Springer, 2009, pp. 225–236.
- [11] A. Endmann, D. Keßner, User journey mapping—a method in user experience design, *i-com* 15 (2016) 105–110.
- [12] I. John, J. Dörr, Elicitation of requirements from user documentation, in: 9th Int. Workshop

- on Requirements Engineering: Foundation for Software Quality (REFSQ), volume 3, 2003, p. 10.
- [13] S. A. Fricker, K. Schneider, F. Fotrousi, C. Thuemmler, Workshop videos for requirements communication, *Requirements Engineering* 21 (2016) 521–552.
 - [14] M. Jirotko, P. Luff, Supporting requirements with video-based analysis, *IEEE Software* 23 (2006) 42–44.
 - [15] N. Banovic, T. Grossman, J. Matejka, G. Fitzmaurice, Waken: reverse engineering usage information and interface structure from software videos, in: *25th Annual ACM Symposium on User Interface Software and Technology*, 2012, pp. 83–92.
 - [16] E. Aghajani, C. Nagy, M. Linares-Vásquez, L. Moreno, G. Bavota, M. Lanza, D. C. Shepherd, Software documentation: the practitioners’ perspective, in: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 590–601.