Business Process Re-engineering and Agile Software Development: Applying the Story-Card Method

Elijah Djan, Marné De Vries

Business Process Re-engineering and Agile Software Development: Applying the Story-Card Method

Elijah Djan[1], Marné de Vries[2][0000-0002-1715-0430]
[1,2] University of Pretoria, Pretoria, South Africa
[1] djanic327@gmail.com
[2] Marne.devries@up.ac.za

Abstract. Enterprise designers need to continuously re-design their enterprise, re-evaluating the technologies that are available to digitize their operations. Although light-weight agile software development approaches are favored by software development service providers, additional requirements elicitation practices should be incorporated when scaling factors apply, since design team members need to have a shared understanding of the operating context and high-level requirements. Research indicated that the organization construction diagram (OCD) could be useful to create a shared context for enterprise operation, linking detailed functional requirements to this shared context during software development. Although the OCD is concise, its associated concepts are abstract and an additional story-card method (SCM) is needed to transform existing enterprise implementations into an OCD. Since additional evaluation of the SCM was required, this study focused on a real-world demonstration of the SCM at a Fintech company where an agile software development approach is applied. The results indicate that the SCM is useful when incorporated within an agile software development approach.

Keywords: Enterprise engineering, story-card method, DEMO, agile software development.

1 Introduction

Within a context of volatility, uncertainty, complexity and ambiguity, enterprise designers need to continuously re-design the enterprise, re-evaluating the technologies that are available to digitize their operations. In accordance with the agile paradigm for information system design, enterprises need iterative design approaches to (re-)design their systems incrementally [1]. During information systems design, agile methods and practices may have to be tailored for contexts where *scaling factors apply*, especially regarding the *elicitation and management of requirements* [2,3].

Since additional *requirements elicitation* practices should be incorporated when *scaling factors apply* [4], the organization construction diagram (OCD), associated with the *design and engineering methodology for organizations* (DEMO), could be used to represent a *blue print* of enterprise operation, creating a foundation for *requirements elicitation* and *tracking* during information systems development [5]. Research indicated that an additional *story-card method* (SCM) was needed to intro-

duce the abstract OCD concepts to agile development stakeholders [5]. Although previous research experimented with the SCM, the method was applied in isolation, i.e. not within a real-world agile software development context. The main contribution of this study is to demonstrate how the SCM is used within an agile software development approach to solve existing deficiencies at a Fintech company.

Next, we briefly introduce the remaining sections of the article. Section 1 elaborates on the OCD, SCM and other methods and practices that were used at the Fintech company. Section 2 introduces *Framework for Evaluation in Design Science (FEDS)* evaluation design process that guided further evaluation of the SCM within a real-world context. In section 3 we present a demonstration of the SCM and in section 4 we present solution implementation- and evaluation results. Finally, we discuss our findings in section 5 and conclude in section 6.

1.1 The SCM to Compile an OCD

The SCM incorporates *collaborative* and *easy-to-use* technologies, i.e. *sticky notes* as *story cards* to facilitate *collaboration* and transformation of existing enterprise implementations (i.e. a *concrete world*) into more *abstract* (and concise) concepts of the OCD [5]. The 5 *inputs* and 10 *method steps* of the SCM is useful to transform a subset of implemented processes into a consolidated view of enterprise operations, depicted by the OCD [5]. We briefly introduce the OCD, i.e. the main deliverable of the SCM.

Dietz (in Perinforma [6]) acknowledges that a user's *needs* for information system support starts with an understanding of their day-to-day operations. He presents *four ontological aspect models* that are *coherent, comprehensive, consistent, and concise* and that are useful to represent the *essence of enterprise operation* [7]. The organization construction model (OCM) is the most essential model and consists of three representations, the *organization construction diagram* (OCD) and the *transaction product table* (TPT) and the *bank content table* (BCT) [6,8].

The OCD provides a graphical representation of actor roles (implemented by human beings) that are involved in a number of transactions. Each transaction is an instance of a particular transaction kind (TK), and every TK incorporates a single production act and multiple coordination acts. Dietz [6,8] indicates that the coordination acts and production act are arranged according to a pattern, called the *complete transaction pattern*. For every TK, two actor roles are involved, one is the initiating actor role and the other one is the executing actor role. Although the executing actor role is responsible to execute the production act, both actor roles perform several coordination acts.

We explain some of these concepts, referring to the OCD that is depicted in Fig. 1. Every OCD represents some essential operations for a particular scope-of-interest. Fig. 1 indicates that the *benefits department* was the scope-of-interest. Furthermore, multiple actor roles may be involved (indicated as rectangles), acting as an initiator (when linked to a solid line) or an executor (when linked to a solid line that ends in a filled diamond). Actor roles interact with one another via several transaction kinds (TKs). Fig. 1 includes two TKs, namely T01 (*claim evaluation*) and T02 (*claim payment*). The diamond-disc representation of the TK indicates that every TK consists of

a production act/fact (represented by a diamond) and multiple coordination acts/facts (represented by a disc). The actor roles are either classified as elementary or composite. An elementary actor role (indicated by a white rectangle) is the executor of only one TK, as exemplified by the *evaluator* in Fig. 1. A composite actor role, indicated by a grey-shaded rectangle, is the executor of more than one TK. By default, actor roles that are outside the scope-of-interest, are modelled as composite actor roles [6]. Therefore, Fig. 1 indicates that the *broker* and *finance department* are composite actor roles. The actor role within the scope-of-interest (i.e. the *evaluator*) may need to have access to production facts and coordination facts that are produced outside the scope-of-interest. Fig. 1 indicates that the *evaluator* needs information access (indicated with a dotted line) to an *in-house software system facts*, i.e. production and coordination facts that are produced outside the scope-of-interest.

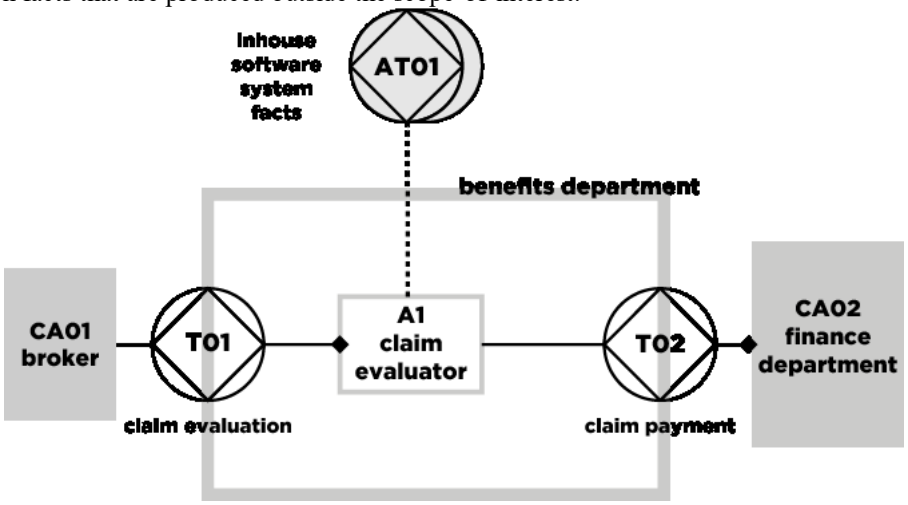

Fig. 1. An OCD for the benefits department

The OCD provides a consolidated view of enterprise operation. Every TK on the OCD follows a transaction pattern. Fig. 2 depicts the complete pattern of behavior that exists between an initiator and an executor when they perform a TK. Thus, the pattern depicted in Fig. 2 applies to T01 with the initiator being the *broker* and the executor being the *evaluator*. Yet, the same pattern is also followed for T02 where the initiator is the *evaluator* and the executor is the *finance department*.

Dietz [6] indicates that a TK may be classified according to three categories. A TK is *original* when the associated production act has the intent of creating an *original/new* fact. A TK is classified as *informational* if its associated production act merely recalls/interprets a previously-created original production facts. Lastly, a TK is *documental* when the associated production fact simply transmits/copies/retrieves/destroys the original fact. Usually, the OCD only represents TKs that are of the *original* sort [6], hence extracting only the essence of enterprise operation.

The OCD presented in Fig. 1 is the main deliverable of the SCM. Yet, the SCM starts by identifying existing operational activities that are performed in a particular se-

quence at an enterprise. Operational activities include TKs of all sorts, i.e. original, informational and documental. The SCM facilitates the process of analyzing the implemented operational activities, re-structuring operational knowledge into a consolidated OCD [5].

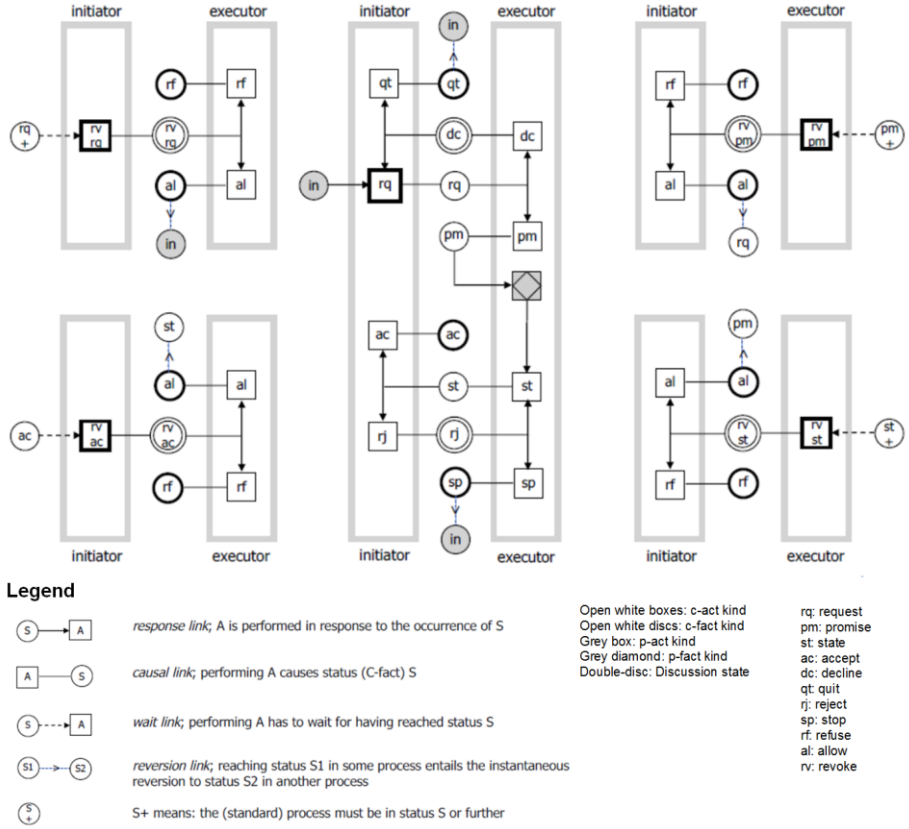

Fig. 2. The complete transaction pattern, from [8]

Once the SCM has been applied, every operational activity: (1) is traceable to a particular TK on the OCD, and (2) can be considered for semi-automation, transforming the activity into a *user story* [5].

A *user story* is a "general-purpose *agile* substitute for what traditionally has been referred to as *software requirements*" [3]. The next section presents a template to elaborate on the software requirements associated with a *user story*.

1.2 User Story Cards and the INSERT criteria

Patel & Ramachandran [9] provided a *template*, called a *user story card*, that is based on the INSERT validation criteria. The INSERT criteria ensures that every requirement is Independent, Negotiable, Small-enough-to-fit-into-iteration; Estimable, Representative-of-user-functionality and Testable. Applying the INSERT criteria, the analyst has to complete ten fields per story card: (1) story card number, (2) project

name, (3) estimation, (4) story name, (5) date, (6) story, (7), acceptance test, (8) note, (9) risk, and (10) points to consider [9]. Later, in section 3.2, we also recommend that another field is added on the *user story card*, tracing each user story back to a TK on the OCD.

2 Research Methodology

The study applied the *Framework for Evaluation in Design Science (FEDS)* evaluation design process [10], applying the four-step process as follows:

Explicate the goals of the evaluation: The main purpose of evaluating the SCM is to determine how well it achieves its expected environmental utility, i.e. using the OCD within an agile system development context within a scaled context to manage and trace requirements.

Choose the evaluation strategy: In terms of the FEDS evaluation strategies, evaluation of the SCM requires iterative episodes of formative evaluation that follows the naturalistic paradigm, since the major risk of the SCM is social or user oriented. According to [10] naturalistic evaluation explores the performance of a solution technology in its real environment, e.g. within a real-world enterprise.

Determine the properties to evaluate: As stated in [5], the SCM's utility could be evaluated in terms of ability to address requirements elicitation criteria for agile development within scaled contexts. One of the utility properties, is the *usefulness* of the SCM within a real-world context.

Design the individual evaluation episode(s): The *first evaluation episode* involved 21 participants to experiment with the SCM, providing survey feedback in terms of its utility [5]. The positive feedback obtained in [5] gave the impetus to experiment further on the *usefulness* of the SCM within a real-world agile software development context as a *second evaluation episode*. The *second evaluation episode* was conducted and presented in this article. We used user acceptance tests and structured interviews to evaluate the *usefulness* of the SCM and to evaluate the newly-developed software solution and its ability to address some of the deficiencies evident at the enterprise. In addition, the participant-observer, experimenting with the SCM, provided some reflections on using and extending the SCM.

Additional evaluation episodes are needed to experiment with the SCM within different *scaled* agile contexts, as indicated in section 6.

3 Story Card Method Demonstration

This section demonstrates the use of the SCM as part of a software development project. Section 3.1 provides background on the company and its problem context. Section 3.2 presents the agile software development approach that incorporated the SCM during the early phases of development, whereas section 3.3 presents an application of the agile software development approach.

3.1 The Enterprise Context

A software development project was initiated by one of South Africa's leading Fintech companies of which the *benefits department* interacts with various *brokers* that claim on behalf of claimants. The problem is that the *benefits department* used MS Excel to track claim evaluations. Since employees used the desktop version of Excel, worksheets could not be accessed simultaneously by the employees and the worksheets contained duplicate entry fields. Rapid growth in business created an urgency for digitizing all manual operations, reducing the lead time for claim payments to beneficiaries, and reducing the number of erroneous claim payments. Even though the benefits department used MS Excel, other departments used an in-house system to automate their processes.

3.2 The Adapted Agile Approach

One of the core values of *agile software development* indicates a preference for individuals and interactions over processes and tools [11]. Tools and techniques should encourage customer collaboration, enable change, ensure working software and minimize the use of documentation [12]. Yet, minimal documentation often creates problems in tracing requirements to their origin, especially when scaling factors apply [13,14].

The software development team, working at the Fintech company, already followed an agile software development approach. An opportunity existed to experiment with the SCM, incorporating it within the existing agile development approach, addressing the deficiencies at the *benefits department*. Similar to the iterative and model-based agile software development approach presented by [15], the software development approach started with a scope definition, followed by two stages: (1) a combined problem analysis, requirements analysis and decision analysis; and (2) multiple iterations of design, construction & testing, review and delivery. The next section demonstrates the adapted agile software development approach, elaborating on the first stage, emphasizing the use of the SCM and *user story cards*.

3.3 Demonstrating the Adapted Agile Approach

This section elaborates on the *problem analysis*, *requirements analysis* and *decision analysis* of the adapted agile software development approach that was used at the Fintech company.

Problem analysis phase. Whetherby's [15] PIECES framework was used to extract information-system related deficiencies in terms of Performance, Information (and Data), Economics, Control (and Security) and Efficiencies. Additional problem analysis was also incorporated as part of the next phase.

Requirements analysis phase. Identified deficiencies resulting from the PIECES analysis, were converted into *non-functional solution requirements*. The *functional requirements* were primarily derived via the SCM. The participant-observer applied the 10-step SCM (see [5]) to extract the existing operational activities from enterprise

participants. We shortly discuss an application of the 10-step method and also indicate two extensions for *Step 1* and *Step 10*.

- *Step 1: Inquire from a colleague to explain a short process (about 10 to 15 activities) that s/he is involved with.* Fig. 3 illustrates the main output of *Step 1*. The participant-observer extended this step, also mapping out the process represented in Fig. 3 to a business process model, using the Business Process Model and Notation (BPMN) specification (see [16]). The extension allowed for critical analysis of the existing process, highlighting inefficient interaction between participants. Problem areas were identified, labelled (e.g. Problem A, Problem B) and described.
- *Step 2: Take a picture (photo) of the process.* See Fig. 3.
- *Step 3: Discuss with your colleague all the actors that are involved and write down composite actors on yellow sticky notes, adding a smiley face, keeping actors aside.* The smiley face sticky notes are illustrated on Fig. 4.
- *Step 4: Explain Dietz's red-green-blue triangle of production acts, also explaining the complete transaction pattern for actor-collaboration regarding production acts.* The participant observer briefly presented the concepts to the participants, only explaining the red-green-blue triangle of production acts. The complete transaction pattern was not explained, since it would be too time-consuming.
- *Step 5: Have a discussion with your colleague as to identify original production acts from his/her process (as mapped out with sticky notes in Step 1).* The discussion and classification followed.
- *Step 6: Classify (in collaboration with your colleague) remaining acts as coordination acts vs. production acts.* Additional classification was performed.
- *Step 7: Remove the original production act notes from the flat surface and phrase appropriate transaction kind descriptions (using adjective+noun) on red sticky notes that are positioned as diamonds on your A1 paper. Collapse initial production act notes underneath re-phrased transaction kind notes.* Fig. 4 illustrates the re-structuring of sticky notes according to the OCD-format.
- *Step 8: The remaining activities on your working space should be coordination acts or informational/documental production acts. Remove each of the remaining notes on your working surface and collapse them underneath the appropriate rephrased transaction kind (red diamond notes) on your A1 paper.* Fig. 4 illustrates the re-structuring of sticky notes according to the OCD-format.
- *Step 9: Position the yellow actor role notes on the A1 paper, drawing in (with a black pen) the initiator actors (+initiating links) as well as the executing actors (+executing links) to the transaction kinds, completing a composite OCD.* Fig. 4 illustrates the re-structuring of sticky notes according to the OCD-format.
- *Step 10: Validate your composite OCD with your colleague.* Validation was performed. The participant-observer extended this step, identifying those activities that had to be automated, using a red dot as illustrated on Fig. 4. The TK with the most red dots should be prioritized and had to form part of the same build cycle.

According to [5], the composite OCD, such as the one depicted in Fig. 4, needs further tailoring as to ensure that *composite actor roles* that form part of the scope-of-

interest are converted into *elementary actor roles*. The final OCD was constructed and is presented in Fig. 1.

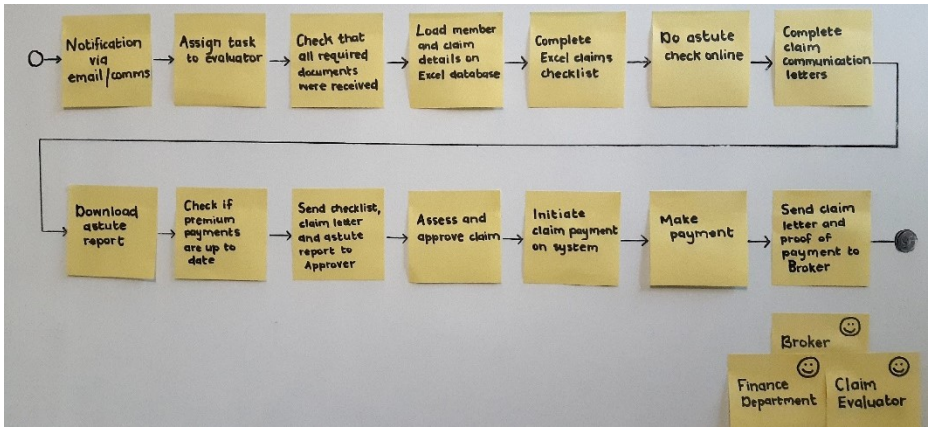

Fig. 3. Example of a process to demonstrate method *Step 1* of the SCM

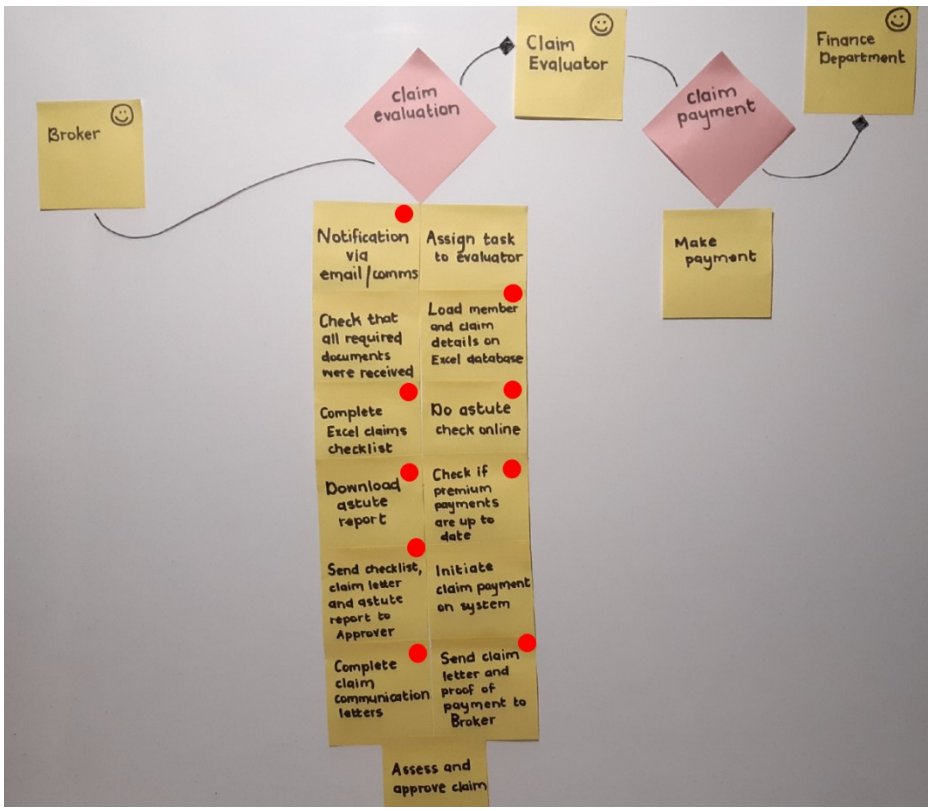

Fig. 4. An OCD with composite actor roles is the deliverables of the SCM

Each of the activities, earmarked for automation, was converted into multiple *user story cards*, adhering to the INSERT criteria and the user story card template presented in section 1.2. The participant-observer also added a field called *traceability*. Each story card had to be traceable to a TK on the OCD (represented in Fig. 1). One of the user story cards, depicted in Fig. 5, linked to a *Problem A* and linked to TK T01 (*claim evaluation*) of Fig. 1.

Story Card 1	**Project Name:** Streamlined Front-End Design	**Estimation:** 1 hour
Story Name: Policy Holder Application Form Upload	**Traceability:** Problem A (T01)	**Date:** 2019/08/29
Story: Policy holders should be able to upload their claim forms and required documents online and not via email.	**Acceptance Test:** 1. Try to upload nothing 2. Try to upload only claim form 3. Try to upload only required documents	
Note: N/A	**Risk:** Low	
Points to be considered: System should be able to check that the correct documents are indeed uploaded via Optical Character Recognition (OCR) and Artificial Intelligence (AI).		

Fig. 5. A user story card linked to TK T01

Decision analysis phase. During this phase, three alternative solutions were identified: (1) Extending the existing in-house software application to replace the existing Excel sheets; (2) Using blockchain technology to create smart contracts to validate and verify the details of the deceased and automate payment; and (3) Creating a web app to replace the existing in-house software application that is currently used by the company. A feasibility analysis matrix, used in accordance with [15], indicated that the urgency of a solution favored the first alternative.

4 Results

Since agile software development approaches value *working software* over comprehensive documentation [11], we had to ensure that the adapted agile approach produced *working software*. Section 4.1 reports on the acceptance test results for four implemented story cards. In addition, we wanted to reflect on the story-card-method and its ease-of-use within a real-world project. Section 4.2 provides some reflections on applying the SCM.

4.1 Acceptance Tests and Interview Results

As indicated in section 3.2 the software development approach consisted of two stages. The second stage consists of multiple iterations of design, construction & testing, review and delivery. For the first iteration, four out of eight story cards were prioritized and used to construct user interfaces. The newly-developed user-interfaces were evaluated by four participants and acceptance tests were completed with positive results. Participants also had to answer five interview questions in terms of a five-point

scale: strongly agree, agree, unsure, disagree and strongly disagree. Participants were encouraged to qualify their response. The results are presented per interview question.

- *Do you believe that the previous process had too many manual steps?* Three participants strongly agreed, while the fourth participant agreed.
- *Has the new solution reduced manual steps by 50%?* Two participants strongly agreed, whereas the other two participants agreed. Participants commented that the consolidation of data capturing on a single system and the use of auto-fill functionality for certain fields drastically decreased the number of manual steps.
- *Do you believe that the new process has reduced processing time of claims?* All participants strongly agreed.
- *Is the new process better than the old process?* Three participants strongly agreed, while the fourth participant agreed. Participants valued the automatic calculation of benefit amounts, replacing the manual entries.
- *Has the use of Microsoft Excel been eradicated?* Three participants strongly agreed, while the fourth participant agreed.

4.2 Reflecting on the SCM

The participant-observer facilitated the SCM and provided some reflections in terms of the 10-step method presented in section 3.3:

- *Steps 1 and 2:* This step was easy to perform, since the participant-observer spent ample time at the company to observe the process. A company participant (the claim evaluator) confirmed the validity of the existing process as mapped out with sticky notes.
- *Step 3:* This step was easy to perform and the company participant willingly provided inputs.
- *Steps 4 to 7:* The company participant was not so interested to participate during the classification and re-structuring of activities, but preferred that the facilitator (i.e. the participant-observer) performed these steps. Still, the participant validated the classifications afterwards.
- *Step 8:* The participant-observer valued this step, since it allows for a consolidated view of existing activities and a means to identify areas of improvement.
- *Step 9:* This step was easy to perform.
- *Step 10:* This step and its extension was easy to perform.

Although the participant-observer performed the SCM with ease, the company participants had difficulty in understanding some of the concepts.

The demonstration of the SCM led to two further extensions. *Step 1* was extended representing the rudimentary sticky-note process flow with a process model according to the BPMN specification, using swim lanes and pools to indicate existing work allocation according to enterprise-specific roles. The extension also allowed for critical analysis of the existing process, especially in terms of *inefficient interaction* between company roles. The detailed process model helped to highlight problem areas that were labelled (e.g. Problem A, Problem B) and described.

A second extension was applied in *Step 10*, namely to use red dots to earmark story-cards for automation. The rationale is that story cards that belong to the same TK and earmarked for automation, should also be built during the same build cycle.

5 Discussion

Enterprise designers need to continuously re-design their enterprise, re-evaluating the technologies that are available to digitize their operations. Although light-weight agile software development approaches are favored by software development service providers, additional requirements elicitation practices should be incorporated when *scaling factors* apply, since design team members need to have a shared understanding of the operating context and high-level requirements. Research indicated that OCD could be useful to create a shared context for enterprise operation, linking detailed functional requirements to this shared context during software development. Although the OCD is concise, its associated concepts are abstract and an additional SCM is needed to transform existing enterprise implementations into an OCD. Since additional evaluation of the SCM was required, this study provided a real-world demonstration of the SCM within an agile software development context.

The SCM was incorporated within an *agile software development approach*. During its application, the enterprise designer (also the participant-observer) identified the need to further extend *Step 1* and *Step 10* of the SCM.

Two different methods were used for evaluation. First, we evaluated the *agile software development approach*, with the embedded SCM, to assess whether the approach rendered *working software*. User acceptance tests indicated positive results. We also used structured interviews, engaging with relevant participants at the benefits department, to assess whether some of the previous information system deficiencies have been addressed. Positive feedback was obtained.

The second evaluation method entailed some reflection on using the SCM. The participant-observer indicated that the SCM was easy to use. Yet, he fast-tracked some of the steps, reducing some of the explanations about OCD concepts. His reflections emphasized the conditional use of the SCM, i.e. that the facilitator needs sufficient knowledge on the theoretical concepts to provide additional explanations where needed.

6 Conclusion and Future Research

The SCM was useful when incorporated within an agile software development approach. Yet, as indicated by the demonstration and participant-observer's feedback, the method may need further adaptation (e.g. extending *Step 1* and *Step 10*) to ensure integration within an existing *agile software development approach*. In addition, the enterprise designer that facilitates the SCM, may need to adapt the theoretical explanations regarding OCD concepts, ensuring that participants will be able to validate the final OCD.

Since the SCM was useful within a real-world context where an agile approach is currently used, *agile at scale* projects, where different scaling factors apply should further validate the usefulness of the SCM within the agile software development context. The main deliverable of the SCM, the OCD, is useful to create a common understanding of the essential operations at an enterprise. The implementation-free OCD becomes the starting point for various different implementation options. Alt-

hough this study favored further development of the in-house information system, the OCD can also be used as the starting point for implementing blockchain technology.

Acknowledgements. This work was demonstrated at a Fintech company. We are grateful for all the assistance and feedback that we received from the company.

References

1. Beck K, Beedle M, Van Bennekum A, Cockburn A et al (2001) Manifesto for Agile Software Development. www.agilemanifesto.org. Accessed 23 April 2018
2. Dikert K, Paasivaara M, Lassenius C (2016) Challenges and success factors for large-scale agile transformations: A systematic literature review. J of Syst and Softw 119:87-108
3. Leffingwell D (2011) Agile software requirements: lean requirements practices for teams, programs, and the enterprise. Addison-Wesley, New Jersey
4. Paasivaara M, Lassenius C (2016) Scaling scrum in a large globally distributed organisation: A case study. In: IEEE 11th International Conference on Global Software Engineering. IEEE Computer Society. doi:DOI 10.1109/ICGSE.2016.34
5. De Vries M (2018) DEMO and the Story-Card Method: Requirements Elicitation for Agile Software Development at Scale. In: Buchmann R, Kirikova M (eds) 11th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modelling. Springer. doi:https://doi.org/10.1007/978-3-030-02302-7
6. Perinforma APC (2017) The essence of organisation. 3rd edn. Sapio, www.sapio.nl
7. Dietz JLG (2006) Enterprise ontology. Springer, Berlin
8. Dietz JLG, Mulder MAT (2017) DEMOSL-3: DEMO specification language version 3.7. SAPIO,
9. Patel C, Ramachandran M (2009) Story card based agile software development. Int J of Hybrid Inf Technol 2 (2):125-140
10. Venable J, Pries-Heje J, Baskerville R (2016) FEDS: A framework for evaluation in design science research. Eur J of Inf Syst 25 (1):77-89
11. The Agile Manifesto (n.d.) Manifesto for Agile Software Development. https://www.agilealliance.org/agile101/the-agile-manifesto/. Accessed 11 November 2019
12. Strode D (2006) Agile methods: a comparative analyis. In: Proceedings of the 19th annual conference of the national advisory committee on computing qualifications, NACCQ.
13. Inayat I, Salim SS, Marczak S, Daneva M et al (2015) A systematic literature review on agile requirements engineering practices and challenges. Comput in Hum Bahav 51 (PB):915-929. doi:10.1016/j.chb.2014.10.046
14. Heikkilä VT, Damian D, Lassenius C, Paasivaara M (2015) A mapping study on requirements engineering in agile software development. In: 2015 41st Euromicro Conference on Software Engineering and Advanced Applications. pp 199-207. doi:10.1109/SEAA.2015.70
15. Bentley LD, Whitten JL (2007) Systems analysis and design for the global enterprise. 7th edn. McGraw-Hill/Irwin, New York
16. Object Management Group (n.d.) Business process model & notation. https://www.omg.org/bpmn/. Accessed 30 May 2019