DEMO and the Story-Card Method: Requirements Elicitation for Agile Software Development at Scale

Marné de Vries [0000-0002-1715-0430]

Department of Industrial and Systems Engineering, University of Pretoria, South Africa
Marne.devries@up.ac.za

Abstract. Enterprises of today are faced with rapidly changing technologies and customer needs within unpredictable environments that require a new mindset for creating an *agile enterprise.* Agile practices gained momentum within software development communities due to their speed-of-delivery and incremental value delivery. Yet, for software development projects *at scale*, theorists believe that stakeholders first need to have a *common understanding* of the *enterprise operational context*, sharing a common *big picture* as part of *requirements elicitation*. The *design and engineering methodology for organizations* (DEMO) encapsulates an organization construction diagram (OCD) that is useful for representing the *enterprise operational context*, i.e. removing unnecessary clutter of technology implementation detail. Theory indicates that abstract OCD concepts are *concise* and used in a *consistent* way. Yet, agile methodologies require models that *encourage collaboration*, are *easy to understand* and relate to a *concrete world*, rather than an *abstract world*. The main contribution of this article is to present a different means of introducing the OCD to software development stakeholders, relating *abstract concepts* of the OCD back to a *concrete world*. Using *design science research*, this study suggests and evaluates a *story-card method* that incorporates collaborative and easy-to-use technologies, i.e. *sticky notes* as *story cards*. Feedback from 21 research participants indicated that the *story-card method* indeed facilitated translation of a *concrete world* into more *abstract* (and concise) concepts of the OCD, also improving the possibility of adopting the OCD at an enterprise as a means to represent a *common understanding* of the *enterprise operational context*.

Keywords: Enterprise engineering, requirements elicitation, organization construction diagram, agile methodologies, agile at scale.

1 Introduction

Most enterprises of today are faced with VUCA (volatility, uncertainty, complexity and ambiguity) and need to operate within unpredictable environments that require a new mindset for creating an *agile enterprise* [1]. Enterprises also need to ensure that they expand their information system landscape in a *dynamic*, but *coherent and integrated way* [2].

Modern software development methodologies have already moved way from the autocratic, plan-driven approaches of the past towards light-weight and agile methodologies that are *iterative and incremental* [3; 4]. A study on agile methods and practices, performed in 2014 by VersionOne [5], inviting 3925 individuals from a broad range of industries in global software development, indicated that 53% of the respondents had *more than 1000 employees* at their enterprise. Since agile software development methods were originally intended for *small and individual teams*, several challenges emerged when agile practices were applied *at scale* [6].

Enterprise *size* is one of many *scaling factors* that need to be considered when adopting an agile methodology at an enterprise. Agile methods and practices may have to be tailored for contexts where *scaling factors apply*, especially regarding the *elicitation and management of requirements* [6; 7]. Since additional *requirements elicitation* practices should be incorporated when *scaling factors apply* [8], we believe that existing methods and practices, associated with the *design and engineering methodology for organizations* (DEMO), could be used to represent a *blue print* of enterprise operation, a foundation for *eliciting requirements* and developing supporting information systems.

In this article we argue that one of the DEMO constructs, called the organization construction diagram (OCD) is useful to communicate the *blue print of enterprise operation*. Yet, agile development stakeholders have different roles and therefor require methods and practices that *encourage collaboration*, are *easy to understand*, and relating to a *concrete world* rather than *abstract concept*s encapsulated in the OCD. Hence, we motivate the need to develop an additional method, called the *story-card method,* to facilitate cognitive understanding of the *abstract concepts* associated with the OCD. The purpose is not to demonstrate how the OCD solves all challenges associated with different kinds of *scaling factors*. Rather, we acknowledge that the OCD will only become useful within agile development contexts if one or more *scaling factors* apply, since more advanced *requirements elicitation and management* is needed when *scaling factors* apply.

Next, we briefly introduce the remaining sections of the article. Section 2 motivates the need to include additional *requirement elicitation practices* within agile methodologies when *scaling factors* apply, also introducing existing theory that may be useful for *requirements elicitation*. Section 3 introduces *design science research* (DSR) as an appropriate research methodology for developing an artefact, the *story-card method*, as a means to incorporate the OCD into agile methodologies when *scaling factors* apply. We present the *story-card method* in section 4 and discuss evaluation results of the *story-card method* in section 5. Finally, we summarize the results in section 6 and suggest opportunities for future research.

2 Background Theory

Agile practices are currently applied to more complex environments than before, creating several challenges, including *requirements elicitation and management* challenges. The purpose of this section is to define the concept *agile at scale*, introducing some of the challenges associated with *agile at scale*. Section 2.1 provides a definition of *agile at scale* and *criteria* for addressing *requirements elicitation and management* challenges associated with projects where *scaling factors* apply. In section 2.2 we present and critique current agile frameworks and practices in terms of the *requirements elicitation criteria,* whereas section 2.3 presents an alternative modelling language, the *design and engineering methodology for organizations* (DEMO) that may be incorporated to address *requirements eliciation criteria*.

2.1 Agile at Scale and the Need for Requirements Engineering

Agile methodologies were originally intended for small teams with collocated team members, working face-to-face in team rooms. Application of agile methods within *scaled* contexts resulted in several challenges regarding coordination between teams (especially for distributed projects), lack of architecture, and *lack of requirement management* [8].

Definition of Agile at Scale and Requirements Elicitation. Different ideas exist on classifying an agile development as *large*, using *project cost, project duration, size of the software developed, number of people, number of teams involved* and *number of sites* [9]. Moe & Dingsøyr [10] believe that scaling should not only be defined in terms of *team size* or *number of teams*, since teams may be distributed across location and enterprise boundaries, creating additional complexity and challenges. Likewise, Ambler & Lines [11] elaborate on different *scaling factors* that may apply: *geographical distribution, team size, regulatory compliance, domain complexity, technical complexity, enterprise distribution, enterprise complexity and enterprise discipline*. In terms of regulatory compliance, regulatory requirements stipulated by Sarbanes-Oxley or BASEL II, may necessitate *documented evidence* for certain processes and *traceability* against relevant standards [12].

Robertson & Robertson [13, p 9] believe that *requirements* "exist either because the type of product *demands* certain functions and qualities, or because the client justifiably *asks for the requirement* to be part of the delivered product". For a software development project, the *product* is a software application. In terms of a software development project, Leffingwell [14] distinguishes between *needs*, *features* and *software requirements*, i.e. different *requirements concepts* that elaborate on end user's operations within an enterprise and how end users expect support from information systems. Section 2.3 elaborates on the need to understand the *organization construction* (i.e. user's operational *needs*), prior to eliciting *features* and *software requirements* for a supporting software application. The *features* and *software requirements* translate *needs* into a software solution. According to Schön et al. [15] *agile* software development still incorporates *requirements elicitation and management*, but in a more iterative way, rather than at the start of the project. *High-level requirements* or operational *needs* should still be defined, but are expanded continuously throughout the project [15]. Yet, when *scaling factors* apply, software development teams need to re-consider the mechanisms and practices that are selected for *requirement elicitation and management* [8].

Understanding the Big Picture for Projects at Scale. According to Schön et al. [15], it is a challenge to keep sight of the *big picture* in terms of the *project vision* for projects where *scaling factors apply*. Schön et al. [15] define a project vision as: "an abstract description of the overarching goal that guides product development and aligns development, business people and other stakeholders". Ambler and Lines [11] believe that the *project vision* should be encapsulated in a number of abstract, *high-level requirements*. They propose several mechanisms for representing high-level requirements, such as a business process model, context diagram, mind map, UI flow diagram, storyboard, value stream maps and UML use case models.

In addition, the entire project team, which may consist of multiple smaller teams, should have a *shared understanding* of the *high-level requirements* [11], as discussed in the next paragraph.

Creating a Shared Understanding. Buchan [16] indicates that customers and software development teams need to develop a common understanding about a client's requirements, since inadequate *sharing and understanding* will have a negative impact on product quality and cost. Buchan [16] analyses the challenges involved in creating a *shared understanding of requirements* (SUR), applying principles from *cognition theory* to address the challenges of obtaining a SUR. He states that SUR is a specialized form of the *team mental model* (TMM) as discussed by Mohammed, Ferzandi and Hamilton [17], i.e. SUR is "viewed as structured mental representations of knowledge and understanding about relevant aspects of requirements, that are similar in each team member" [16]. The content of SUR is *shared knowledge structures* that include declarative (what), procedural (how) and strategic (why) knowledge about requirements [18]. Furthermore, the property *shared* indicates that team members have some common or overlapping (but not identical) knowledge structures that are *consistent* [19]. Lastly, SUR has the property of "accuracy", since it has to be aligned with the true state of the world [20]. A gap in SUR may indicate that relevant knowledge about a requirement is: (1) missing, or (2) lacks sufficient detail, or (3) is not adequately shared between team members, or (4) is inconsistent between team members, or (5) is an error, i.e. inconsistent with the *concrete world*. Project team members need to first *communicate* or *share* their ideas, making them explicit in the form of representations, such as *narratives or models*, before they could reach consensus on a *shared understanding* [21; 22].

Creating Traceability. Minimal documentation, as promoted by *agile methodologies*, creates problems in *tracing* requirements to their origin [23; 24]. Even though traceability may be perceived as a heavy-weight activity with little value, distributed projects still obtain more benefits than incurring costs [25]. Leffingwell [7] presents a traceability model to indicate how different kinds of models communicate requirements and how the models are related.

2.2 Existing Agile Frameworks and Embedded Mechanisms and Practices

Addressing *scaling* challenges, several *scaling frameworks* were developed, such as the Scaled Agile Framework (SAFe), Large-Scale Scrum (LeSS), Disciplined Agile Delivery (DAD), Scrum of Scrums (SoS) and LeanSAFE [26]. Two of the five frameworks addresses the *lack of requirements elicitation and management* challenge, identified by Paasivaara & Lassenius [8], a challenge that enterprises face, when they adopt *agile* within a context where *scaling factors apply*.

Leffingwell's [7] SAFe suggests *three levels of scaling*: (1) agile team level, (2) program level and (3) portfolio level. Each level suggests a *minimum number of artefacts, roles and practices* for effective software product delivery.

Ambler's [27] DAD provides a different means of *scaling*, already discussed in section 2.1, and he provides a practice-based methodology that focuses on *effective modelling and documentation* of software products. DAD incorporates mechanisms from Scrum, Extreme Programming (XP), Agile Modelling (AM), Unified Process (UP), Agile Data (AD) and Kanban. DAD provides an Agile Scaling Model (ASM) as a foundation for scaling agile mechanisms according to the enterprise context, *without being too prescriptive* on the requirements modelling and documentation practices [27]. He also provides three scaling levels: (1) core agile development; (2) agile delivery; and (3) agile at scale. For all three levels, the level of detail of the *initial requirements* models and descriptions will differ depending on the type of project with the purpose of doing *just enough requirements elicitation* to gain agreement on the scope of the project [27]. Ambler [27] suggests that the development team uses modelling mechanisms that are inclusive, such as *drawing diagrams on white boards*.

Existing agile frameworks/methodologies follow a pragmatic approach, in suggesting *easy-to-use* modelling mechanisms and practices that are appropriate for the project context [21]. Patton & Economy [21] also indicate that agile practices, such as *user stories* and *user story mapping*, already address two of the three *requirements elicitation criteria*, discussed in section 2.1. They argue that *user stories* and *user story mapping* can be used to (1) represent the *operational context* (i.e. the *big picture*) and (2) a *shared understanding* of requirements. The subsequent sections provide more detail about *user stories* and *user story mapping*, evaluating their ability to consolidate requirements *in a consistent way* into a *big picture*.

User Stories. User stories are the "general-purpose agile substitute for what traditionally has been referred to as *software requirements*" [7, p 37].

User stories intend to relate to the *concrete world* of a user. Since a user story is framed as a goal, goals may encapsulate many sub-goals and the constructional complexity required to achieve a goal need not be stated when defining a goal. Patton & Economy [21] use the analogy of a rock (i.e. a goal) that is broken into pebbles (i.e. sub-goals). The problem is that there is no consistency in how the rock is broken into different-sized pebbles. Trkman, Mendling & Krisper [28] warn against the use of user stories, since there may be a *lack of dependencies* between user stories and their relationship to the overall context. In dealing with the last-mentioned problem, Patton & Economy [21] suggests that user stories are *mapped*.

User Story Mapping. User story mapping seemingly satisfies the *requirements elicitation criteria* of providing a *shared understanding* and a *big picture* representation. The idea is that *user stories* are mapped on a large working space (e.g. office wall) to indicate relationships between stories, but also decide on priorities for software development [21]. The *user stories* are mapped as steps on *sticky notes*, sequencing from left to right, also discussing the software support that would be needed per step and placed vertically below the relevant step [21].

Even though *user stories* and *user story mapping* address two of the *requirements elicitation criteria*, namely a representation of the *big picture* and using easy-to-use *sticky notes* to create a *shared understanding*, requirements encapsulated in the user stories are not consolidated *in a consistent way* into higher-level stories. Although *use case models and narratives* provide more structure for software requirements detail than *user stories*, requirements detail encapsulated within use cases are also based on goal-decomposition [29]. In the next section, we present an alternative to *user stories* or *use cases*, namely the identification of *transaction kinds* that provide a consistent means for consolidating enterprise operational detail. In addition, we indicate how the *transaction kind* is used as part of an organization construction diagram (OCD).

2.3 The Organization Construction Diagram (OCD)

Background on the OCD. Similar to Leffingwell [7], Dietz (in Perinforma [30]) 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* [31]. The organization construction model (OCM) is the most essential model and consists of two representations, the *organization construction* ~~*model*~~ *diagram* (OCD) and the *transaction product table* (TPT) [30].

The OCD provides a graphical representation of *actor roles* (implemented by human beings) that perform a number of *coordination acts* (e.g. requests and promises) with regards to *production acts*. The *production acts* may be either immaterial (e.g. devising, deciding or judging) or material (e.g. manufacturing or transporting) [30]. Furthermore, *production acts* may be classified as *original* (e.g. devising, deciding or judging), *informational* (e.g. recalling, deriving or calculating), or *documental* (e.g. saving, retrieving, copying, transmitting or destroying) [30]. Yet, *original* production acts are supported by *informational* production acts, which are in turn supported by *documental* production acts. Many software applications are developed as technologies to semi-automate or implement some of the *coordination acts* and the *production acts* [31].

Dietz [30] argues that software development stakeholders need to have a *common understanding* of the *original* production acts, since other acts (informational and documental) and implementation technologies (software applications) merely support the *original* production acts. Focusing on the *original* production acts, it is possible to compile a *concise* representation or *big picture* of the *operational context*.

Fig. 1 provides a graphical representation of an OCD that consists of four *original production acts* and four *actor roles*, based on the following narrative: "Every year, in consultation with the CEO, the enterprise designer selects members for an enterprise governance committee, capturing the selected members on our *enterprise design application* (EDA). The selected members should also indicate their willingness to become members of the committee. Later, the enterprise designer refers back to the information about selected members to request from every selected member to participate at a workshop. The purpose of the workshop (a periodic event) is that the entire committee needs to evaluate enterprise governance concepts. When committee members arrive at the workshop, the enterprise designer first ensures that all members state their participation by signing an attendance register before the workshop can start. The workshop assistant also captures the attendance data on EDA. The selected committee members often become involved in other projects and then need to resign from the committee. In that case, the enterprise designer consults/communicates with the CEO to replace the committee member, i.e. re-select a member."

Partially explaining the constructs of Fig. 1, the actor role, *annual member selector*, initiates (via the solid link) *annual member selection*. The same actor role, *annual member selector*, is also the executor of *annual member selection* (represented via the solid link that ends in a solid diamond). Furthermore, the same actor role, *annual member selector*, also initiates *committee membership starting*. The combined diamond-disk constructs on Fig. 1 are called *transaction kinds*. Each *transaction kind* incorporates a *production act/fact* (represented via a diamond) as well as multiple *coordination acts/facts* (represented via a disc). Often, the generated *facts* need to be shared with other actor roles, since the facts may have an effect on the actor's behavior. Thus, additional information links (dotted links) are used to indicate access to particular coordination and production *facts*. For instance, the actor role in Fig. 1, *governance concepts evaluator*, needs to have access to the facts that are generated by *committee-membership starting*, since the *governance concept evaluator* has to involve members that already committed themselves to become members of the committee.

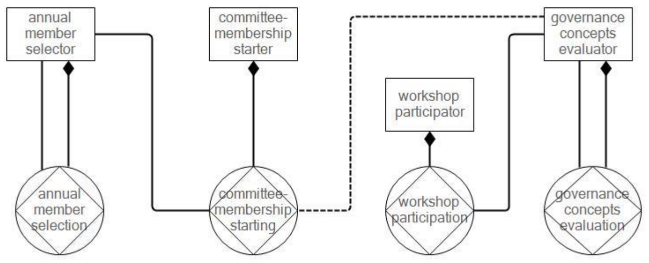
annual member selector
committee-membership starter
governance concepts evaluator
workshop participator
annual member selection
committee-membership starting
workshop participation
governance concepts evaluation

Fig. 1. Elementary OCD modelled with ABACUS

Addressing Requirements Elicitation Criteria with the OCD. In section 2.1 we already indicated that additional *requirements elicitation* practices are only required within *agile methodologies*, if *scaling factors* apply. In addition, we presented *three criteria* for *requirements elicitation* practices. We now motivate that the OCD has the potential to address the *three criteria,* i.e. (1) representing the *big picture,* (2) creating a *shared understanding* of the big picture, and (3) providing sufficient structure to ensure *traceability of requirements*.

As discussed in the previous section, we believe that software development stakeholders (including enterprise stakeholders) need to have a *shared understanding* of the *original* production acts, since other production acts (informational and documental) and implementation technologies (software applications) merely support the *original* production acts. The OCD provides a *concise* representation or *big picture* of the *operational context* in a *consistent way*, i.e. every *transaction kind* (diamond-disc) on the OCD, represents an *entire transaction pattern* of an *original* production act and multiple coordination acts [30].

Regarding *traceability of requirements*, the structural composition of four *aspect models* (i.e. the organization construction model (OCM), the process model (PM), the action model (AM) and the fact model (FM), already ensure integration and traceability between the *four aspect models*. In accordance with Leffingwell's [14] distinction of requirements into *needs*, *features* and *software requirements*, we believe that the *four aspect models* provide sufficient structure to *trace software requirements* back to operation-supporting *needs*. As an example, the user may need software application support for the transaction kind *annual member selection* (see Fig. 1), i.e. to capture information about the selected members on a software application system called EDA. We acknowledge that other *features* may also be required from EDA, such as the ability *log onto the EDA system*. The stated *feature* will however not be traceable to a particular *transaction kind*.

Although the OCD has the potential to address the three *requirements elicitation criteria* that we identified in section 2.1, *agile methodologies* require modelling techniques and tools that *encourage collaboration*, are *easy to understand* and have the ability to relate back to a *concrete world* [21]. The next section presents a research methodology for designing a *story-card method* to facilitate reference to a *concrete world* and the *ease of understanding* OCD concepts.

3 Research Methodology

The study applied *design science research* (DSR), developing a new artefact, namely a *story-card method*, to enhance *ease-of-understanding* of OCD concepts when the OCD is used for requirements elicitation. According to Gregor & Hevner's [32] knowledge contribution framework, the *story-card method* can be considered as an improvement, since the *method* will be used for solving a known problem. Referring to the DSR steps of Peffers et al. [33], this article addresses the *five steps* of the DSR cycle in the following way:

Identify a problem: Previous research highlighted that agile methodologies are useful for small projects, but may require additional *requirements elicitation* practices for projects or enterprises where *scaling factors apply* [7]. As discussed in section 2.3, the OCD has the ability to represent the *essence of enterprise operations* in a *consistent way* and the potential to convey a *shared understanding* of the enterprise *operation context*, also called the *big picture*. The *problem* is that agile development stakeholders have different roles and therefor require methods and practices that *encourage collaboration*, are *easy to understand*, and relating to a *concrete world* rather than *abstract concept*s encapsulated in the OCD.

Define objectives of the solution: Acknowledging the potential of the OCD to create a *shared understanding* of the enterprise *operation context*, whilst addressing the problem that agile team members require methods and practices that are *collaborative*, *easy to understand* and relate to a *concrete world*, an additional *method* is required to enhance understanding of the OCD concepts.

Design and development: In accordance with the solution objectives, a new method, i.e. the *story-card method*, was designed to introduce OCD concepts to *participants* from different backgrounds, i.e. addressing the need to create a *shared understanding* amongst stakeholders that fulfil *different roles*.

Demonstration: The *story-card method* was demonstrated to industry participants during an interactive session. During the demonstration, participants had the opportunity to criticize the *method*. The feedback was also used to refine the *story-card method*.

Evaluation: The industry participants evaluated the refined *story-card method* in practice by involving a colleague. A questionnaire, consisting of 18 questions/probes, was used to evaluate whether the *story-card method* addressed the solution objectives. In addition, the participants had to reflect whether modelling with *sticky notes* is preferred rather than using software modelling tools. Lastly, the participants had to obtain feedback from their colleague on whether the colleague would be confident to use the *story-card method* in future.

4 The Story Card Method

The *story-card method* specifies 5 *inputs* and 10 *method steps*.
Inputs: (1) flat working space, such as table or white board, (2) A1 paper, (3) sticky notes of 2 different colors (red and yellow), (4) a black pen, (5) a colleague's inputs.
Method steps:

- *Step 1:* Inquire from a colleague to explain a short process (about 10 to 15 activities) that s/he is involved with. Ensure that the process incorporates the use of information technology (e.g. the process followed from requesting vacation leave up to receiving notification about the approval of the request). Explain to your colleague that s/he needs to write the tasks (verb+noun) on yellow sticky notes and position the notes in sequence of occurrence, left to right on a flat working space (e.g. desk or white board).
- *Step 2:* Take a picture (photo) of the process. [Note that this step was only inserted to ensure that participants provided evidence about the initial process].
- *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.
- *Step 4:* Explain Dietz's red-green-blue triangle of production acts, also explaining the universal transaction pattern for actor-collaboration regarding production acts.
- *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 4).
- *Step 6:* Classify (in collaboration with your colleague) remaining acts as coordination acts vs. production acts.
- *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.

- *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 re-phrased *transaction kind* (red diamond notes) on your A1 paper.
- *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*.
- *Step 10:* Validate your *composite OCD* with your colleague.

The method steps were demonstrated to the participants. Fig. 2 represents the result for performing *Steps 1 to 3*, whereas Fig. 3 resulted from performing *Steps 4 to 10*.

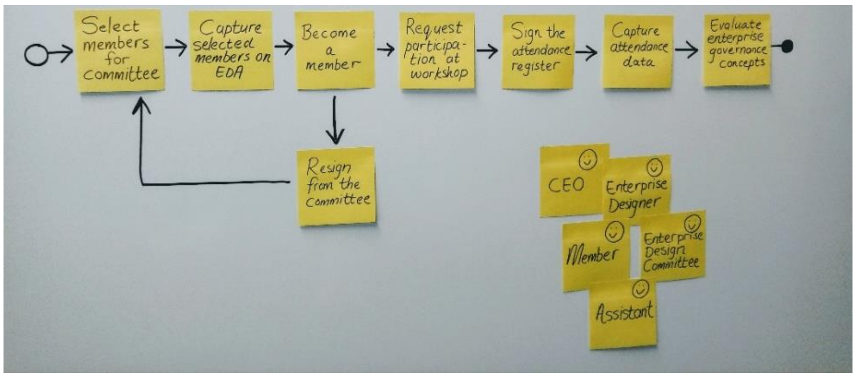
Select members for Committee
Capture selected members on EDA
Become a member
Request participa-tion at workshop
Sign the attendance register
Capture attendance data
Evaluate enterprise governance concepts
Resign from the Committee
CEO
Enterprise Designer
Member
Enterprise Design Committee
Assistant

Fig. 2. Example of a process to demonstrate *method step 1* of the *story-card method*

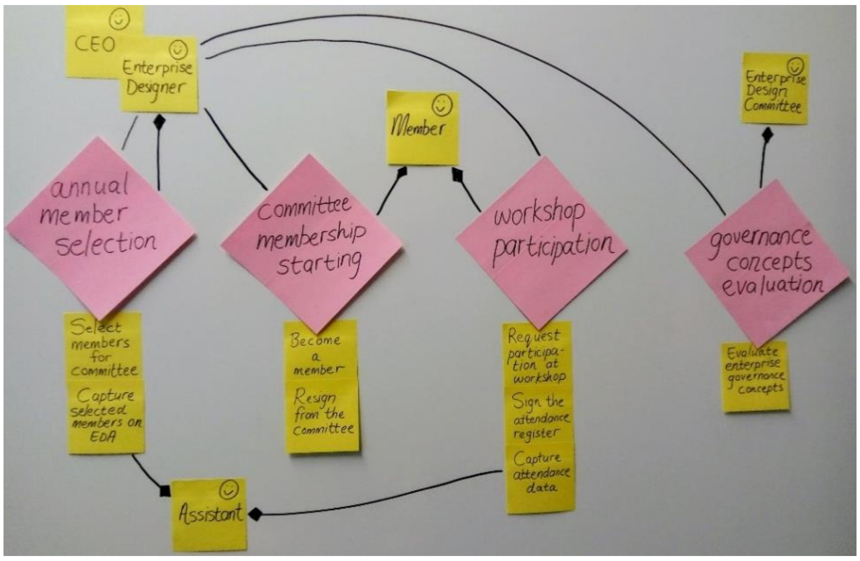
CEO
Enterprise Designer
Member
Enterprise Design Committee
annual member selection
Committee membership starting
workshop participation
governance concepts evaluation
Select members for Committee
Capture selected members on EDA
Become a member
Resign from the Committee
Request participa-tion at workshop
Sign the attendance register
Capture attendance data
Evaluate enterprise governance concepts
Assistant

Fig. 3. An *OCD with composites* is the main deliverable of the *story-card method*

The *OCD with composites* would require additional work as to be transformed into an *OCD withe elementary actor roles*. Thus, referring to Fig. 3, the yellow sticky notes at the bottom of the diamond-shaped *transaction kinds* need to be removed from the diagram, whereas the *composite actor roles* positioned above the diamond-shaped *transaction kinds* need to be replaced with *elementary actor roles*. An *elementary OCD* has been compiled using the software ABACUS, presented in Fig. 1.

5 Results

Although 32 participants applied the *story-card method*, only 21 participants completed the voluntary survey. The following sub-sections synthesize the questionnaire results.

5.1 Participant Background

Responding to the question "*Indicate your existing role at the enterprise*", 25% of the responding participants (4 out of 20) are business analysts, whereas the remaining participants represented 12 various different roles. Different industries were represented from both the manufacturing, services and consulting sectors (i.e. aerospace and defense manufacturing, automotive, construction, education, financial services, software vendors, industrial manufacturing, mining, agricultural, consulting and travel/transportation). Most of the participants have an Industrial Engineering background (i.e. 10 BEng and 3 BTech Industrial Engineering participants), whereas the remaining participants have an Engineering background (Mechanical Engineering, Metallurgical, Mining Engineering and Chemical) or a Science (BSc) background. The 10 BEng participants also have software development background, since a module (Information Systems Design) forms part of their undergraduate curriculum.

5.2 Feedback on the Story-Card Method

Participants had to indicate the time duration for completing the 10 *story-card method* steps. The average time to complete was 103 minutes, with a median of 105 minutes. Other descriptive statistics (the large standard deviation of 68 minutes, the minimum of 30 minutes and maximum of 300 minutes) indicate a huge variation when the *story-card method* is applied.

As motivated in section 2.3, the OCD already has the ability to address three *requirements elicitation criteria* for scaled contexts, i.e. (1) understanding the *big picture*, (2) creating a *shared understanding* and (3) *traceability*. The *problem* is that agile team members require methods and practices that are *collaborative, easy to understand* and relate to a *concrete world* and the *story-card method* was developed to enhance understanding of OCD concepts. Thus, participants had to evaluate whether the solution artefact, i.e. the *story-card method*, was *useful* in relating to a *concrete world* when explaining abstract concepts of the OCD to a colleague. Feedback was positive. Participants (20 out of 21) that answered the question of whether *the story-card method helped to relate process steps to OCD constructs*, either agreed (14 out of 20) or strongly agreed (6 out of 20). In addition, participants either strongly agreed

(7 out of 21), agreed (12 out of 21) or were neutral (2 out of 21) when they responded to the question on whether *the story-card method encouraged discussion with a colleague to classify appropriate activities as original production activities versus informational or documental production activities*. Participants were also positive to use the *story-card method* in future to explain OCD concepts, i.e. they either strongly agreed (9 out of 21), agreed (11 out of 21) or were neutral (1 out of 21) that *if I had to explain OCD concepts to another colleague in future, I would use the story-card method, rather than my own/another way of explanation*.

Evaluating whether participants *preferred to use sticky notes* (exemplified in Fig. 2 and Fig. 3) rather than using modelling software, such as ABACUS (exemplified in Fig. 1) for modelling, most participants (15 out of 21) *preferred manual modelling with sticky notes*, whereas some (6 out of 21) preferred to use *modelling software*. Participants had to motivate their preferred modelling method.

Themes extracted from those that *preferred manual modelling with sticky notes* include: Easy to understand (4 responses); encourages conversation/discussion (4 responses); less intimidating for the interviewee (1 response); less stressful to the modeler (1 response); allows for changes by moving sticky notes around (1 response); and sticky note modelling is useful for initial modelling (1 response). Yet, 2 responses indicate that the participants selected the manual option, since they never had exposure to software modelling tools before.

Themes extracted from those that preferred to use *software modelling* tools include: Easier to implement changes (3 responses); easier to draw the sequential process (1 response); sticky notes do not always stick (1 response); and diagram readability is improved (1 response).

Referring to section 4 (i.e. the *method steps*) participants had to indicate whether they *experienced any difficulties in using the story-card method*. A number of themes emerged from the 8 responses:

- Step 3: Preferring a swim-lane diagram to assign actor roles to process steps (1 response).
- Step 5: Difficulty in deciding whether an activity is an *original* production activity (2 responses).
- Step 7: Difficulty in changing the sticky note descriptions from verb+noun to adjective+noun (2 responses).
- Step 9: Difficulty in explaining the purpose of the red diamond (1 response).
- Step 10: Difficult to validate and confirm the OCD with the company representative (1 response).
- Not step related: Difficulty in obtaining participation from Step 4 onwards, since non-technical staff "zoned out" when new concepts were introduced (1 response).

Evaluating whether participants *consider using the OCD within their own working environment*, rendered positive results, since 15 (out of 21) participants either strongly agreed (4 out of 21) or agreed (11 out of 21). Only one participant disagreed, providing the following rationale: *"I do not think it would add much value to my current working environment. The value that it would add is not worth the difficulty of explaining to someone the technicalities of the OCD (personal opinion)."* Two participants that were neutral (neither agree nor disagree) indicated that *"Although it is not my preferred method of process modelling, it was still useful to map out the process in that manner. It provided a different aspect of the process"* and *"It's not used a lot in my work domain. However, if I had been in the enterprise engineering field, I would consider using it"*.

The *story-card method* had to ensure *ease-of-understanding*, relating *abstract concepts* of the OCD to a *concrete world*. In accordance, we evaluated whether the colleagues *would be confident to use the story-card method to model another process by him/herself to construct a composite OCD.* Almost half of the *colleagues* agreed (10 out of 21), some responded neutral (7 out of 21) and a few (4 out of 21) disagreed. Three of the *colleagues that disagreed* indicated that they would need another example prior to applying the *story-card method* with confidence. The other *colleague that disagreed* indicated that he/she did not understand the theory behind the story-card method and had difficulty in identifying the different type of acts.

Finally, participants had to present an *elementary OCD* (i.e. Figure 5.2 from Perinforma [30, p 74]) to their *colleagues* to inquire whether *a similar kind of diagram would be useful to represent a blue print of their enterprise operations.* The intension was to evaluate whether the OCD could be adopted as a means for representing a *big picture* for *essential enterprise operations*. The responses were overall positive, ranging from strongly agreeing (1 out of 21), agreeing (10 out of 21) and being neutral (10 out of 21).

6 Discussion and Future Research

For software development projects *at scale*, stakeholders first need to have a *common understanding* of the *enterprise operational context*, sharing a common *big picture* as part of *requirements elicitation*. The *design and engineering methodology for organizations* (DEMO) encapsulates an organization construction diagram (OCD) that is useful for representing the *enterprise operational context*, i.e. removing unnecessary clutter of technology implementation detail. Theory indicates that that abstract OCD concepts are *concise* and used in a *consistent* way. Yet, agile methodologies require models that *encourage collaboration*, are *easy to understand* and relate to a *concrete world*, rather than an *abstract world*.

This article presented a different means of introducing the OCD to software development stakeholders, relating *abstract concepts* of the OCD back to a *concrete world*. Using DSR, this study suggested and evaluated a *story-card method* that incorporates *collaborative* and *easy-to-use* technologies, i.e. *sticky notes* as *story cards*. Feedback from 21 research participants indicated that the *story-card method* indeed facilitated *collaboration* and translation of a *concrete world* into more *abstract* (and concise) concepts of the OCD. The *story-card method* also improved the possibility of adopting the OCD at an enterprise as a means to represent a *common understanding* of the *enterprise operational context*. Since participants represented various different industries and roles, we believe that the *story-card method* would be useful within various different contexts, including context where *scaling factors* apply.

The qualitative feedback obtained from participants regarding difficulties experienced in applying some of the *method steps* provide the opportunity for further improvement of the *story-card method*. In addition, the *story-card method* may also need additional development to ensure its own scalability. We suggest that the *story-card method* is applied within real-world *agile at scale* projects where different scaling factors apply as to further validate the usefulness of the *story-card method* and the OCD within software development projects. Since this article did not expand too much on the *traceability* criterion, we suggest that Leffingwell's [7] meta-model for requirements concepts is adapted to demonstrate *traceability* of requirements when the *four aspect models* are included as part of the meta-model.

Acknowledgements

This work is based on the research supported wholly / in part by the National Research Foundation of South Africa (Grant Number 115089). We are also grateful towards the research participants for their valuable feedback.

References

1 Meyer, P.: Agility shift: Creating agile and effective leaders, teams, and organizations (1st ed.). Routledge, New York (2016).

2 Hoogervorst, J. A. P.: Practicing enterprise governance and enterprise engineering - applying the employee-centric theory of organization. Springer, Berlin Heidelberg (2018).

3 West, D., Grant, T., Gerush, M., D'silva, D.: Agile development: Mainstream adoption has changed agility. Forrester Research, 2(1), 41 (2010).

4 Zucker, A.: What we really know about successful projects, https://www.scrumalliance.org/community/articles/2016/october/what-we-really-know-about-successful-projects, last accessed 2018/14 May.

5 VersionOne: 9th Annual State of Agile Survey 2015, https://www.watermarklearning.com/downloads/state-of-agile-development-survey.pdf, last accessed 2018/6 June.

6 Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: A systematic literature review. Journal of Systems and Software, 119, 87-108 (2016).

7 Leffingwell, D.: Agile software requirements: lean requirements practices for teams, programs, and the enterprise. Addison-Wesley, New Jersey (2011).

8 Paasivaara, M., Lassenius, C.: Scaling scrum in a large globally distributed organisation: A case study. In: IEEE 11th International Conference on Global Software Engineering. IEEE Computer Society (2016).

9 Dingsøyr, T., Moe, N. B.: Research challenges in large-scale agile software development. ACM SIGSOFT Software Engineering Notes, 38(5), 38-39 (2013).

10 Moe, N. B., Dingsøyr, T.: Emerging research themes and updated research agenda for large-scale agile development: a summary of the 5th international workshop at XP2017. In: Proceedings of the XP2017 Scientific Workshops. pp. 1-4. ACM, Cologne, Germany (2017).

11 Ambler, S. W., Lines, M.: Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise. IBM Press, US (2012).

12 Ambler, S. W.: Agile software development at scale. In: Meyer B., Nawrocki J.R., Walter B. (eds.) Balancing Agility and Formalism in Software Engineering. Lecture Notes in Computer Science. Vol. 5082. Pringer, Berlin, Heidelberg (2008).

13 Roberson, S., Roberson, J.: Mastering the requirements process: Getting requirements right (3rd ed.). Addison-Wesley, New York (2013).

14 Leffingwell, D.: Scaling software agility. Pearson Education, Boston (2007).

15 Schön, E., Thomaschewski, J., Escalona, M. J.: Agile requirements engineering: A systematic literature review. Computer Standards & Interfaces, 49, 79-91 (2017).

16 Buchan, J.: An empirical cognitive model of the development of shared understanding of requirements. Requirements Engineering, 432, 165-179 (2014).

17 Mohammed, S., Ferzandi, L., Hamilton, K.: Metaphor no more: A 15-year review of the team mental model construct. Journal of Management, 36, 876-910 (2010).

18 Rouse, W. B., Cannon-Bowers, J. A., Salas, E.: The role of mental models in team performance in complex systems. Systems, Man and Cybernetics, IEEE Transactions, 22, 1296-1308 (1992).

19 Cannon-Bowers, J. A., Salas, E., Converse, S. A.: Shared mental models in expert team decision making. In: Castellan, J. (ed.) Current Issues in Individual and Group Decision Making. pp. 221-246. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ (1993).

20 Edwards, B. D., Day, E. A., Arthur, W. J., Bell, S. T.: Relationships among team ability composition, team mental models, and team performance. Journal of Applied Psychology, 91, 727-736 (2006).

21 Patton, J., Economy, P.: User story mapping: discover the whole story, build the right product. O'Reilly Media Inc., Sebastopol (2014).

22 Kannan, V., Fish, J. C., Willett, D. L.: Agile model driven development of electronic health record-based specialty population registries. In: 2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI). pp. 465-468. (2016).

23 Inayat, I., Salim, S. S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Computers in Human Bahavior, 51(PB), 915-929 (2015).

24 Heikkilä, V. T., Damian, D., Lassenius, C., Paasivaara, M.: A mapping study on requirements engineering in agile software development. In: 2015 41st Euromicro Conference on Software Engineering and Advanced Applications. pp. 199-207. (2015).

25 Cleland-Huang, J.: Traceability in agile projects. In: Cleland-Huang J. et al (ed.) Software and Systems Traceability. pp. 265-275. Pringer-Verlag, London (2012).

26 Ebert, C., Paasivaara, M.: Scaling Agile. IEEE Software, 34(6), 98-103 (2017).

27 Ambler, S. W.: The disciplined agile (DA) framework: a foundation for business agility, http://www.disciplinedagiledelivery.com/, last accessed 2018/6 June.

28 Trkman, M., Mendling, J., Krisper, M.: Using business process models to better understand the dependencies among user stories. Information and Software Technology, 71, 58-76 (2016).

29 Cockburn, A.: Writing effective use cases. Addison-Wesley, Indianapolis (2001).

30 Perinforma, A. P. C.: The essence of organisation (3rd ed.). Sapio, www.sapio.nl (2017).

31 Dietz, J. L. G.: Enterprise ontology. Springer, Berlin (2006).

32 Gregor, S., Hevner, A.: Positioning and presenting design science research for maximum impact. MIS Quarterly, 37(2), 337-355 (2013).

33 Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S.: A design science research methodology for information systems research. Journal of MIS, 24(3), 45-77 (2008).