The Journal of Systems and Software 113 (2016) 418–429

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Aligning codependent Scrum teams to enable fast business value delivery: A governance framework and set of intervention actions

Jan Vlietland[a], Rini van Solingen[b], Hans van Vliet[c,*]

[a] Search4Solutions B.V., Professional Services, Utrecht, The Netherlands
[b] Department of Electronics, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands
[c] Department of Computer Science, VU University Amsterdam, Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 27 September 2014
Revised 6 October 2015
Accepted 7 November 2015
Available online 19 November 2015

Keywords:
Agile
Chain codependencies
Collaboration
Coordination
Alignment

ABSTRACT

Many enterprises that adopt Agile/Scrum suffer from collaboration issues between Scrum teams that depend on one another to deliver end-to-end functionality. These dependencies delay delivery and as a result deteriorate the business value delivered in such value chains. The objective of our study is to support enterprises that suffer from such dependencies with a governance framework that helps them mitigate collaboration issues between sets of codependent Scrum teams. We first identify a set of intervention actions that aim to mitigate the collaboration issues between codependent Scrum teams. Second, we validate the effectiveness of these intervention actions in a large confirmatory industrial case study. This study was held in a large multi-national financial institute that worked with a large number of codependent Scrum teams. Third, we triangulate the findings in three focus groups. We finally package the intervention actions in a governance framework. The intervention actions led to a delivery time reduction from 29 days to 10 days. The participants in the focus groups confirmed the causality between the intervention actions and the observed delivery improvement. The empirical results show that the intervention actions, packaged in the governance framework, enable codependent sets of Scrum teams to deliver faster.

1. Introduction

Large companies operating in information intensive industries experience rapid changing business demands, requiring swift adaption of front to back (business) value chains. Since these value chains are automated with IT services, the rapid changing business demands require flexible IT services. The IT services that enable these front to back value chains are delivered by a portfolio of interdependent applications. That application portfolio is typically delivered by multiple codependent IT service providers (ISP). IT service changes therefore often require software development staff of multiple ISPs (Plugge & Janssen, 2009; TFSC, 2011). These ISPs have to jointly execute a fast paced software development process (Moniruzzaman & Hossain, 2013; Pikkarainen et al., 2005).

To achieve a fast paced software development process, many internal IT development centers increasingly transfer to Agile methods. The most common Agile method used in industry is the Scrum software development method (VersionOne, 2013). Scrum is an incremental method that uses low boundary cross-functional collaboration in software development teams that work toward a set team goal (Sutherland & Schwaber, 2013). Scrum works with fixed iterations of less than one calendar month that deliver working and tested increments, resulting in faster delivery.

Scrum teams can be mapped in different ways onto the (interdependent) application portfolio. Some prefer a single Scrum team for all interdependent applications that support the front to back value chain (Sutherland, 2005). Two constraints make such coverage difficult. Firstly, the number of involved IT staff (typically from different ISPs) then easily exceeds the generally agreed upon maximum Scrum development team size of 9 members. Secondly, changes typically require highly specialized skills (due to a complex IT landscape with multiple commercial-off-the-shelf items) that cannot be easily shared within a single team. The solution chosen in companies is to set up dedicated Scrum teams. Each Scrum team then develops one or more applications in the portfolio that automates a part of the front to back value chain (Vlietland & van Vliet, 2015). Together, the applications developed by multiple Scrum teams result in value-adding features. Features are defined as 'intentional distinguishing characteristics of the application landscape that can be used by a business user' (IEEE, 2008), e.g. a mortgage registration feature.

* Corresponding author.
E-mail addresses: j.vlietland@Search4Solutions.nl (J. Vlietland), d.m.vansolingen@tudelft.nl (R. van Solingen), hans@cs.vu.nl (H. van Vliet).

http://dx.doi.org/10.1016/j.jss.2015.11.010
0164-1212/© 2015 Elsevier Inc. All rights reserved.

Since features are the result of codependent software development activities by multiple Scrum teams, collaboration between the teams is needed. Particularly the high frequency of deliveries common in Scrum settings makes collaboration a performance factor (Dorairaj et al., 2012). Yet, due to the nature of Scrum teams, such collaboration might not happen naturally. A Scrum team has specific characteristics, such as a maximum of 9 members, a multidisciplinary setup, allocated IT applications, high-frequency deliveries and focus on a single product backlog (Sutherland & Schwaber, 2013). These characteristics typically limit the focus of a Scrum team, resulting in collaboration issues (Vlietland & van Vliet, 2015).

Vlietland and van Vliet (2015) identified six blocking issues in sets of codependent Scrum teams. In the current study, the next step is taken. A set of intervention actions (IAs) for sets of codependent Scrum teams to support a front to back value chain is developed. The IAs aim to mitigate the blocking issues, reducing the delivery time of new features by the set of Scrum teams. The IAs are solidly embedded in organizational change and performance improvement intervention literature.

The IAs are validated in a large confirmatory case study with a set of codependent Scrum teams at a multinational financial institute, during a period of 9 months. After deploying the IAs the delivery time was reduced from 29 days to 10 days. The effect of the IAs, measured in this case-study, was triangulated in focus groups consisting of the members of the codependent Scrum teams. These focus group sessions confirmed that the observed reduction was a direct result of the IAs. The results indicate the effectiveness of the IAs. The IAs were subsequently packaged into the Scrum Value chain Framework (SVF) to support practice in deploying the IAs.

The remainder of this article is organized as follows. Section 2 covers the related work for developing the IAs and the related work regarding (Agile) governance frameworks. Section 3 develops the intervention actions (IAs). Section 4 provides an overview of the empirical results. Section 5 discusses the results and presents the Scrum Value chain Framework (SVF). Section 6 summarizes the threats to validity. Section 7 concludes the study, deduces implications and suggests future research avenues.

2. Related work

Three areas of related work are studied. First, an overview of organizational change literature is given, to position the research design and the IAs in Section 3. Second, the Agile IA literature related to intended performance improvements is studied, to extract the IAs from the Agile literature in Section 3. The section closes with related work on Agile governance frameworks to develop the Scrum Value chain Framework (SVF) and validate the completeness of the IAs in Section 5.2.

2.1. Organizational change literature

Three perspectives on change in the organizational change literature are identified: (1) the tempo of change, (2) planned versus spontaneous change and (3) top-down versus bottom-up change. After introducing these three perspectives, a deeper analysis is performed on a combination that fits this case study, while introducing learning theory as catalyst for organizational change. The section closes with a summary of the change design for this case study.

Tempo of change: One perspective on organizational change is the tempo of change (Weick & Quinn, 1999). At one end of the spectrum is evolutionary change, which involves a relatively long stream of small changes in reaction to the changing environment, as first modeled by Darwin. Evolutionary change in organizations progresses continuously. Revolutionary change at the other end of the spectrum happens in short bursts (Hannan & Freeman, 1984). One theory in the area of revolutionary change is the theory of inertia and punctuated equilibrium (Romanelli & Tushman, 1994). In case an organization does not evolutionary follow the changing environment, the organization gets disconnected from the environment and tends to an inert equilibrium state (Gersick, 1991). In such a state it is hard to change the organization. After some time, strategic reorientation is required to realign the organization with the environment, resulting in a revolutionary change. For such revolutionary change the inert equilibrium needs to be punctuated. After the inertia is broken, the organization experiences a turbulent change to find a new equilibrium, closer aligned with the environment.

Planned versus spontaneous change: A related perspective to evolutionary and revolutionary change is planned versus spontaneous change. Spontaneous change occurs without a set purpose. Each individual actor interacts with other actors and the system changes through evolution (Stacey, 1995). At the other end there is planned change. The actors together aim to achieve a planned state.

Top-down versus bottom up: A perspective related to planned change is top-down versus bottom-up change. Yamakami (2013) analyzed organizational change initiatives in the IT industry and identifies three types of initiatives (1) top-down, in which top management takes initiative, (2) bottom-up, in which the work floor staff takes own initiatives to realize change, and (3) a hybrid approach.

Deeper analysis: Cummings and Worley (2014) elaborate on planned change as a way to change organizations. They identify two planned change strategies (1) a positivistic approach with an unfreezing, moving and freezing phase and an (2) interpretivistic approach with iterations and feedback loops (Jrad et al., 2014). Positivistic based change paradigms have long dominated the IT industry, such as CMMI (Team, 2010) and ISO 9000 (Hoyle, 2001). The positivist paradigm uses a machine metaphor in which input is transformed to output (Ilgen et al., 2005; Stelzer & Mellis, 1998). The paradigm stimulated the use of detailed prescribed work processes which can be quantitatively measured, analyzed and controlled (Unterkalmsteiner et al., 2012). A positivistic approach works in areas of high predictability. The intrinsic human intensive activity of software development with high levels of unpredictability and uncertainty however seems a misfit with such a positivistic paradigm (Clarke & O'Connor, 2013). That misfit was answered at the beginning of this century when the interpretivistic based Agile paradigm got momentum (Akbar et al., 2011). The Agile paradigm uses a bottom-up, continuous change paradigm to utilize human capital in the software development industry (Van Tiem, et al., 2006). Agile is supported with iterations and feedback loops to increase the evolutionary change capability (Qumer & Henderson-Sellers, 2008). Baskerville and Wood-Harper (1996) and Baskerville (1999) specify cyclical action research based on the description of Susman and Evered (1978). Their research design consists of five phases: diagnosing, action planning, action taking, evaluating and specifying learning. The five phases are repeatedly executed to allow adaptation of the change strategy during each cycle.

Learning as catalyst: Experience-based learning can be seen as catalyst for organizational change in Agile environments. Kolb (1984) uses three models of experiential learning for developing a model that combines experience, perception, cognition and behavior. His experience learning model consists of four phases: (1) concrete experience, (2) reflective observation, (3) abstract conceptualization and (4) active experimentation. The capacity to reflect on past experience is one of the key principles for continuous learning in Agile environments (Holz & Melnik, 2004; Salo & Abrahamsson, 2005). Such reflective practice exists in different development disciplines on individual, team and organizational level. For instance a Scrum team conducts a demo and notices that the Product owner repeatedly struggles with a drag and drop action. Such observation allows the team to rethink the functionality and experiment another solution. Qumer and Henderson-Sellers (2008) similarly argue that an agile knowledge engineering and management approach should be integrated with an agile software development approach, and be used for performance improvement, learning and decision making in an agile software development environment.

Change design: This case study fits an evolutionary intervention strategy while having a planned objective. The IAs are designed to achieve that objective. Given the Agile characteristics it is expected that a hybrid, iterative change approach fits the purpose of the case study. The research design is further elaborated in Section 4.

2.2. Agile performance improvement intervention literature

In this section the Agile performance improvement intervention literature is discussed, for each of the five collaboration related issues: coordination, prioritization, alignment, automation and visibility (Vlietland & van Vliet, 2015).

Coordination: The Scrum of Scrums is a Scrum practice to coordinate collaboration between Scrum teams. That practice comes with challenges. Paasivaara et al. (2012) show that Scrum of Scrums works poorly in case of too many participants with disjoint interests. A way to further coordinate work is by using product teams. Schnitter and Mackert (2011) outline how Scrum was scaled with liaison relations between Scrum teams, by introducing product teams that are each responsible for up to seven Scrum teams. The characteristics of such a product team are that each member of the product team is a member of a Scrum team and that each product team bears full responsibility (time, cost and result). Kniberg and Ivarsson (2012) report the implementation of a two level structure combined with liaison relations between Scrum teams to coordinate collaboration, similar to a matrix organization. Scheerer et al. (2014) introduce a more conceptual multi-team system perspective with three types of coordination: (1) mechanistic coordination – with plans, rules and programming, (2) organic coordination – with mutual adjustment and feedback and (3) cognitive coordination – by means of similarity configuration. Product teams utilize such coordination, for instance by making plans and rules and responding to feedback (Vlietland & van Vliet, 2014b).

Prioritization: Another way to improve collaboration between Scrum teams is to prioritize the work over the Scrum teams (Stettina & Hörz, 2015). The literature about priority matching between backlogs is scarce. Rautiainen et al. (2011) study the introduction of portfolio management to support scaled Agile development, by prioritizing all projects in a single backlog. Prioritization dramatically reduced the number of ongoing projects, enabling visibility about ongoing projects that assisted coordination. The product teams of Schnitter and Mackert (2011) are one way to match backlogs of codependent Scrum teams, in case product owners are linked to the product teams. A way to determine which backlog items need to be prioritized over the Scrum teams is explained by Vlaanderen et al. (2011). They introduce a software product management (SPM) process of managing requirements, defining releases and defining products with many stakeholders.

Alignment: The Literature about the alignment of Scrum teams is scarce as well. The literature study did not reveal literature that describes alignment interventions. In Scheerer et al. (2014), alignment is embedded in coordination. Mechanistic alignment of Scrum teams can be achieved by implementing plans and rules similar to those promoted by Leffingwell (2007). Leffingwell (2007) promotes an aligned sprint heartbeat and mentions a define/build/test workflow for all teams. Organic and cognitive alignment is achieved with a shared mental model (Jonker et al., 2011; Lim & Klein, 2006; Mathieu et al., 2000). Shared mental models are implemented by grouping people together and stimulate communication and feedback, such as with the Scrum of Scrums practice. Mechanistic alignment prescribes the alignment practices, while organic and cognitive alignment actually embeds these practices between teams.

Visibility: For the visibility intervention literature, the Agile and Supply Chain Management research areas were studied. Vacanti and Vallet (2014) explain the IAs at Siemens to shift from traditional Agile metrics to actionable flow metrics. Selecting and visualizing flow metrics opened the way to even greater agility, improving predictability and improving performance. The identified IAs are: (1) defining key goals with key metrics and (2) clearly visualizing these metrics such as cycle times, including predictions of future cycle times. Supply Chain visibility has (Scrum) value chain related characteristics. Banbury et al. (2010) explored the role of collaboration between teams by simulating a supply chain and studying the resulting bullwhip effect. The bullwhip effect leads to a drop in productivity in a chain of suppliers, due to a combination of change in demand and a delayed response to that change. The results show that team focused groups need information about the current demand level in the supply chain to minimize the cost, back-orders and bullwhip size and maximize the delivery of orders. Bartlett et al. (2007) investigate the link between visibility and business performance by implementing enhanced visibility of plans, materials and inventory management. Vlietland and van Vliet (2014b) studied the effect of visibility of past performance information onto the actual performance of IT incident handling. Their case study revealed that such visibility has a positive effect on IT incident handling performance.

Automation: In the area of automation of IT processes, the information technology for information technology (IT4IT) literature was identified that describe implementation practices, although none of the papers mentions a value chain supported by a set of codependent Scrum teams. Olsson et al. (2012) present a multiple case study on the move from traditional development to continuous delivery. They identified that during the implementation, collaboration and information exchange is poorly supported and old conservative technology restricts the automation of software development practices. Humble and Farley (2010) describe various practices for the implementation of continuous integration, testing and deployment, by focusing on the technical implementation aspects. Neely and Stolt (2013) report their experience with the implementation of continuous delivery practices. Their approach is to use an evolutionary approach and gradually decrease the delivery time with one step at the time, in line with an evolutionary change approach.

2.3. Agile governance framework literature

This section provides an overview of the Agile governance framework (AGF) literature. Brown and Grant (2005) classify governance as: *"Systematically determining who makes each type of decision (a decision right), who has input to a decision (an input right) and how these people (or groups) are held accountable for their role".* They add that a governance framework should make clear: (1) who has decision making authority, (2) who provides input about a decision and (3) how these roles are jointly held accountable. We identified the following set of AGF core-elements in related work: (1) Role, (2) Event, (3) Team, (4) Artifactand (5) Lifecycle.

These AGF core-elements are used in Section 3 to validate the completeness of the set of IAs. The remainder of the section is structured along the set of identified AGF core-elements.

Role: The Scrum framework includes three roles: Product Owner, Scrum Master, and other Scrum team member. A Product Owner acts as the single 'voice of the customer', collecting and prioritizing customer needs onto a prioritized list of items, the product backlog. The Scrum Master facilitates the Scrum team in achieving its goal. The Scrum team has the responsibility to develop software based on the Sprint Backlog (Rising & Janoff, 2000; Sutherland & Schwaber, 2013). Larman and Vodde (2013) introduce an area product owner as additional role in Agile development to coordinate multiple product owners. Roles with clear responsibilities and authority are therefore identified as AGF core-element.

Event: Sprint Planning, Daily Scrums and a Sprint Review are team events of the Scrum method (Sutherland & Schwaber, 2013) that support self-organization (Moe et al., 2008). Larman and Vodde (2013) introduce an augmented framework for large scale Agile development. The augmentation addresses coordination needs by additional events that support cross team coordination: (1) inter-team Sprint Planning meetings, (2) inter-team Daily Scrums, (3) inter-team Product Refinements and (4) inter-team Sprint Reviews.

Team: The development team in Scrum has a small size (max 9). The small team size eases intra-team knowledge sharing and utilizing the self-organizing ability in professional teams (Takeuchi & Nonaka, 1986). Also Ambler (2009) defines team size as a core-element when Scrum is scaled. Larman and Vodde (2013) use feature teams and liaison relations with Communities of Practice for exchanging knowledge and coordinate between teams. Schnitter and Mackert (2011) identified product teams (similar to feature teams) for managing interdependencies between Scrum teams. Based on the related work, small teams (up to 9 members) are identified as AGF core-element.

Artifact: Self-organizing practices within Scrum teams are supported by artifacts, such as a Product Backlog with everything that is needed in a product, and a sprint backlog with product backlog items selected for a sprint (Sutherland & Schwaber, 2013). Leffingwell (2007) and Leffingwell (2010) promote a three level framework for a portfolio structure of stories, features and epics (cluster of features). Artifacts are therefore identified as AGF core-element.

Lifecycle: A Scrum development lifecycle normally consists of short (2–4 weeks) iterations, which enables swift feedback from software users and related stakeholders about the developed solution. Soundararajan and Arthur (2009) use two phases in their framework for large scale systems: (1) a generation process to gather requirements and (2) a scaling development process for large scale systems. Hence lifecycle is identified as AGF core-element.

3. Research method

This section explains the setup of the confirmatory case study. The case study is performed in a large multi-national financial institute delivering financial services to multinational business customers. The institute uses Agile throughout their complete IT organization, and at the time of our study it had over 300 Scrum teams. The case-study concerns a set of six codependent Scrum teams that are lined up to support one specific value chain.. The case study has five phases, following Runeson and Höst (2009): (1) Designing the case study and designing the interventions; (2) preparing for data collection; (3) collecting evidence; (4) analysis of collected data; and (5) reporting. This section is organized in that order.

3.1. Case study design

A confirmatory case study setup is selected (Easterbrook et al. 2008) to test the impact of the IAs onto the cycle time of feature stories delivered by the Scrum teams. One could argue that reusing an existing (traditional) framework, such as CMMI or ITIL (Team, 2010; van Bon et al. 2007) is the way forward. Agile/Scrum however is based on a philosophy that finds its roots in social constructionism and interpretivism science philosophies (Walsham, 1995). The intervention approach in this case study is aligned with that philosophy, using the perceived issues as departure point. Given the Agile philosophy (Akbar et al., 2011; Qumer & Henderson-Sellers, 2008), a planned, evolutionary intervention approach is chosen (Weick & Quinn, 1999).

Each IA is top-down planned and initiated (Yamakami, 2013). The top-down initiation aims to break the existing equilibrium within the organization (Romanelli & Tushman, 1994). Each IA is designed in a way that multiple members are stimulated to iteratively adapt, refine and further deploy the IA, after the top-down initiation. The iterative approach is enabled with learning (Kolb, 1984) and aims to deeply embed the organizational change.

Archival records are studied to identify the sociological effects of the interventions onto the people operating in the set of codependent Scrum teams. These effects act as rationale for adapting and refining the IAs (Kolb, 1984). Focus group interviews at the end of the intervention period triangulate the effect of the IAs onto the cycle time.

Based on the scaling factors of Ambler (2009), selection criteria are defined for developing the applicable case study selection criteria, as shown in Table 1. The item between brackets at the end of each description refers to the scaling factors.

The selection criteria enable the identification of the unique characteristics of a set of Scrum teams that support the front to back value chain and enhance the content validity of the research. Each Scrum team needs to be experienced and work in accordance with the Scrum framework to minimize research bias.

3.2. Intervention action design

Vlietland and van Vliet (2015) identified six issues in chains of codependent Scrum teams: (1) mismatches in backlog priority between teams, (2) a lack of coordination in the chain, (3) alignment issues between teams, (4) a lack of IT chain process automation, (5) a lack of information visibility in the chain and (6) delivery unpredictability. This section explains the initially designed IAs for mitigating these issues, except for unpredictability. Unpredictability directly impacts the cycle time of new features and is considered the dependent variable of the IAs, following Vlietland and van Vliet (2015).

The design of the intervention actions is based on the related work in Section 2. To mitigate a lack of commitment for top-down IAs (Scheerer et al., 2014) top-down and bottom-up interventions actions are combined in a hybrid implementation approach, as identified by Yamakami (2013).

The related work of Section 2.1 and 2.2 is used to theoretically explain the expected effect of the IAs. Each of the IAs impacts the IT workflow processes of the codependent Scrum teams. Each IA results in deployable 'IA items' that are indicated in **bold** (e.g. **feature description**).

Section 2.3 identified five AGF core-elements. Our aim is to cover all AGF core elements by the IA items. A reference to the core-element is indicated by bold brackets '**<...>**'). The IAs are categorized according to the identified collaboration related issues in Vlietland and van Vliet (2015).

3.2.1. Issue 1: prioritization

IA: Multiple Scrum teams collaborate to jointly deliver added-value features. Each feature will be described in a **feature description <artifact>**. These feature descriptions are broken down in stories on the Product Backlog of each Scrum team that supports the value chain. Each feature description includes the added value and high-level effort estimation. A feature description consists of a functional feature description and a technical interaction design.

IA: The feature analysis and design activities incorporate many uncertainties and can therefore hardly be estimated in one week. For this reason Soundararajan and Arthur (2009) is followed by defining two lifecycle phases: (1) a preparation phase that prepares features, the **Flow to Ready (F2R) <lifecycle>,** and (2) an execution phase that realizes the features, the **Iterate to Done <lifecycle>**. Feature design and analysis activities take place in the F2R phase that takes 'N' weeks to accomplish (Vlaanderen et al., 2011).

IA: Each feature is prioritized on the **feature backlog <artifact>** to match the story priority on the Product Backlog of each Scrum team that supports the value chain. Prioritization will be based on added value and effort. Each feature is described by a feature description consisting of a functional feature description and a technical interaction design. The prioritization mechanism is similar to the

Table 1
Case study selection criteria.

Selection criterion	Selection criteria description
Application interdependencies	Applications have stable interdependencies with other applications in the front to back value chain (technical complexity, domain complexity).
Chain setup	Scrum teams support a front to back value chain and each application under development is allocated to one Scrum team (organizational distribution, organizational complexity, and technical complexity).
Application experience	Each Scrum team develops each of the allocated applications for at least 6 months (technical complexity, organizational complexity and enterprise discipline)
Team distribution	Studied Scrum team members are working in the same country (geographical distribution).
Regulatory requirements	Non user requirements exist that must be taken into account by the product owners (regulatory compliance)
Culture	Studied Scrum team members have the same nationality (organizational complexity)
Agile transition state	Each of the teams acts in a Scrum setup for at least 6 months (organizational complexity).
Workflow automation	Scrum teams already use a single database to manage the development workflow (organizational complexity).

mechanism of Rautiainen et al. (2011). Rautiainen et al. (2011) describe the prioritization of a portfolio of projects, while in this study a portfolio of features on a feature backlog is prioritized (see **Error! Reference source not found.**, feature backlog and the matching arrows to the Scrum team backlogs). Unique feature priority likely also mitigates disjoint interests during Scrum of Scrums (Paasivaara et al., 2012).

IA: Next to the Scrum team Product Owners (PO) that already exist, **Feature Product Owners (FPO) <role>** will be allocated. A Feature Product Owner owns the functionality of a set of (front to back) features.

IA: An **Epic Product Owner (EPO) <role>** will be allocated, being accountable for the unique priority of each feature on the Feature Backlog.

IA: All three Product Owner types in scope of the codependent Scrum teams will be part of the **Product Owner Group (POG) <team>**. The POG discusses and decides about the priority of each feature on the feature backlog. The POG is headed by the Epic Product Owner.

IA: The Product Owner group will meet weekly during the **Epic Planning <event>**. Subgroups of Product Owners will meet regularly on an as needed basis to prepare the priority in the weekly meeting. These interacting groups and subgroups of product owners will enable the forming of a shared mental model (Jonker et al., 2011). It is reasonable to expect that such a shared mental model combined with a clear responsibility will stimulate matched priority settings.

3.2.2. Issue 2: coordination

IA:Product teams <team> crossing the Scrum teams will be set up to coordinate the work between the Scrum teams, as outlined by Schnitter and Mackert (2011) and Kniberg and Ivarsson (2012). Product teams consist of product owners, IT architects, functional analysts and interface designers. A product team will be headed by a feature Product Owner. Typical multiple concurrent product teams exist.

A product team elaborates a feature into a feature description that can be broken down into stories. Product teams have similarities with the system teams of Leffingwell (2010). The functional analysts and interface designers work part-time in a Scrum Team and part-time in Product Teams.

IA: Product teams will meet in **Bi-daily Features <event>** for sharing results, and discussing next actions and impediments. Compared to Kniberg and Ivarsson (2012) product teams focus on sprint preparation activities preventing unnecessary dependencies rather than managing such dependencies during the sprint.

IA: Feature Planning <event> meetings will be scheduled that precede the sprint planning of the Scrum teams. During a feature planning meeting the elaborated features will be used by the Scrum teams to determine and estimate the team specific stories.

IA: A **Scrum of Scrums <event>**will be implemented to organically manage codependencies between the Scrum teams. The Scrum of Scrums will be facilitated by a Scrum coach to secure the effectiveness of the meeting and prevent the issues as identified by (Paasivaara et al., 2012). The Scrum of Scrums will be executed weekly.

IA: Interface connectivity between two applications developed by different Scrum teams is enabled by middleware and interface-adapters. The middleware and adapters are developed by a third dedicated Scrum team. For each interface, therefore, three Scrum teams are involved. **Mini Scrums <team>**centered on interface connectivity will be setup with an analyst/designer from each Scrum team to develop the interface designs and dependency coordination. These Mini Scrums mitigate the issue of disinterest in the Scrum of Scrums as identified by Paasivaara et al. (2012). The **Mini Scrum <event>**take place bi-daily to weekly, depending on the need. The Mini Scrums are facilitated by a Scrum Master (SM) of one of the Scrum teams.

IA: During the **Feature Review <event>**the functionality that was developed by the codependent set of Scrum teams is demonstrated. The Feature review will be scheduled by the Epic Product Owner and facilitated by the applicable Feature Product Owners.

IA: During the **Feature Retrospective <event>**the Product team will evaluate the sprint and plan improvements to be enacted during the next sprint.

3.2.3. Issue 3: alignment

IA: A four week **Aligned Sprint Lifecycle <lifecycle>** duration will be institutionalized over all Scrum teams that support the value chain to make sure that each team delivers stories within the same expected time-frame, being mechanistic alignment (Scheerer et al., 2014). The sprint duration will be institutionalized via the management team and then implemented via the Product Owners and Scrum Masters to the Scrum teams.

IA: The **Aligned Sprint Start <lifecycle>** will align the sprint heartbeat. All Scrum teams work toward the same point in time, the feature review. A fully aligned sprint heartbeat ensures natural alignment in activities between Scrum teams.

IA: A common workflow over all teams will be rolled-out, consisting of predefined workflow states for feature, stories and tasks. Feature, stories and tasks each have their applicable, standardized workflow. Common workflow enables the advantages of a shared mental model as described by (Jonker et al., 2011).

A story with a 'Ready' state will indicate a story that can be picked up for the sprint planning meeting, the state 'Todo' will indicate a story accepted by the Scrum team for a sprint. The ready definition will be bullet wise written down as the **Aligned Definition of Ready (DoR) <artifact>**. Features, stories and interface designs will be developed until the Definition of Ready is met by the product team.

IA: A story with the 'Done' state will be the indication for a story that can be demonstrated in a feature review. At that time the story has been realized and system tested, including interfacing and middleware testing. To allow full understanding of the status of a story before the 'Done' stage is reached, the stories test cycle will also be aligned between the teams. Such elaborated **Aligned Definition of Done (DoD) <artifact>** will align the shared understanding (Jonker et al., 2011) between the Scrum teams, helping teams to adapt and mitigate possible delays of other teams.

3.2.4. Issue 4: automation (IT4IT)

IA: A **Workflow Application <artifact>** will be deployed to support the feature development lifecycle. Each feature will be entered into the application and tagged with a unique priority. The underlying stories will also be entered in the application and linked to the feature. Entering and updating the features and the feature workflow status will fall under the responsibility of the product team. Each Scrum team will be responsible for entering and updating the stories and the story workflow state. The development tasks will be entered into the application and linked to a story by the Scrum team.

3.2.5. Issue 5: visibility

IA: The Workflow Application will support the feature development lifecycle and will enhance visibility over the structure with features, stories and tasks. On each level, planning, status, progress and impediments will be visualized for progress tracking of each item in the lifecycle. For instance the application is able to send an e-mail to each user in the set of Scrum teams in case a story or feature changes state or priority. All information in the workflow application will be accessible by all members. Compact minutes of meeting will be created and shared with the stakeholders. The prioritized list of features will also be shared with all Product Owners, Scrum Masters and IT managers on a weekly basis. Collaboration will be improved by enhanced visibility about the new way of working, as confirmed in the studies of Bartlett et al. (2007) and Vlietland and van Vliet (2014b). Actively sharing the new way of working with all Product Owners, Scrum Masters and IT managers will help punctuating the equilibrium of the existing organizational state (Gersick, 1991).

3.3. Preparing for data collection and collecting evidence

The mail application and archive is used to collect the responses as a result of the IAs. Collected information of each response are the response date, the responding person and the content of the response, with attachments if existent.

The data about story cycle time is extracted from the company workflow application (database). The database contains the stories (issue type Story) per Scrum team, which are exported to Excel with the workflow application web-end. For each of the involved Scrum teams the status change timestamps are extracted from the database with a customized MySQL query and a Putty terminal. The timestamps *TodoDate* and *DoneDate*, as defined in Table 2, are collected.

Table 2
Variables and description of the cycle time variables.

Variable	Description
TodoDate	Timestamp that a story was committed in a sprint for the first time.
DoneDate	Timestamp that a story was moved to the done state

Table 3
Mail database with typical keywords.

Variables	Typical keywords
Coordination	Feature, Owner, Master, Coach, Scrum of Scrums
Prioritization	Group, Priority, Excel, Progress, Daily
Alignment	Definition, Status, Sprint, Time, Date, Workflow
Automation	<Workflow Application Name>
Visibility	<Status update>, Attachments, Minutes

After analyzing the mail application and archive, and calculating the reductions on cycle-time, focus groups are setup to determine the impact of the interventions onto the cycle time, from the perception of the Scrum team members. The focus groups aim to validate the causality between the observed improvements and the IAs, instead of other interventions, actions or influencing factors. Focus groups have been found useful for generating information and shedding light on data already collected, and can be used prior, during and after events or experiences (Krueger & Casey, 2008). The focus groups in this study will evaluate the impact of the performance improvement after the IAs have been performed. Focus groups with four to six participants organized as small groups are more comfortable for the participants, as we expect some level of existing discomfort due to reorganization (Krueger & Casey, 2008; Morgan et al., 2002). The expectation is that these (mini) focus groups deliver more in-depth results, as participants likely have a great deal to share and the discussed topic has a high complexity (Krueger & Casey, 2008). The participants of each group are homogeneously selected to stimulate a focused discussion.

We neutralized bias by splitting the focus group session in two parts. The first part identified and refined the focus group categories independent from the IAs and the second part determined the impact of each identified category. During the first part the top 10 from each individual was collected before integrating them into categories, thus preventing influence of dominant individuals in the focus groups. The influence of the focus group setup on the confirmations of the IAs is therefore considered to be low.

The interventions and activities that – according to the focus group – had the most impact on the shorter cycle time are collected and quantified using post-its. The focus groups also discuss and categorize the post-it items for improved contextual understanding of the post-it items. Each focus group session is audio recorded to reduce analysis bias.

A focus group with Product Owners and a focus group with Scrum Masters of the codependent teams were compiled, since these roles are directly involved in coordination, prioritization and alignment activities. A focus group with Feature product owners was compiled, since these are actors that perform mechanistic front to back coordination (Scheerer et al., 2014) .

3.4. Analysis of collected data

The start, duration and content of the IAs were determined by analyzing the mail history and find typical keywords such as shown in Table 3. Each of the mail is analyzed for key responses (keywords) as result of an intervention.

The average cycle time of feature stories (ASD) is determined per week for each codependent team (T) by calculating the average

Table 4
Performance variables.

Variable	Description	Metric
T	Team identifier in the set of codependent Scrum teams	$T \in 1-6$
O	Week number of the week that a story is opened	$O \in$ Week number
C	Week number of the week that a story is closed	$C \in$ Week number
$N(C, T)$	Amount of stories for team (T), closed in week (C)	$N \in 0-$ m stories
$n(C, T)$	Story identifier for team (T) closed in a week (C)	$n \in 1-N$
SD (n)	Amount of days that story (n) is open	DoneDate (n) – TodoDate (n)
ASD (C,T,n)	Average number of open days for all stories by team (T) closed in week (C)	$\sum_{n=1}^{N} SD_{(C,T,n)}/N$
$\text{ASD}_{\text{start}}$	First measurement week of the average number of open days	4 weeks after start IAs
ASD_{end}	Last measurement week of the average number of open days	6 months after start IAs

number of open days (SD) of the feature stories (S) that were closed in that week (C). The overview of the performance variables is shown in Table 4. The performance analysis was done once after the IAs were deployed. For the total average cycle time of feature stories of all teams the measurement starts 6 weeks after the start of the interventions.

The post-it items of the focus groups are analyzed to triangulate the IAs and the performance improvement. Each of the post-it items is categorized. These categories are compared with the collaboration related issues. The audio recordings are used as point of reference. Post-it items that cannot be translated onto the collaboration related issues are kept under the categories as defined by the focus groups. The sum of the allocated IA points (during step 3) determines the quantitative impact of a category.

4. Results

4.1. The case

A case in the banking industry at a large multinational bank which delivers financial services to large business customers was subject of study. The case entails a set of Scrum teams that offers solution delivery services to a high-volume banking value chain at the multinational bank. The case conforms to the selection criteria of Table 1.

The value chain is supported by six Scrum teams with technical interdependencies between the applications under development by the Scrum teams. The front-office application (developed by Scrum team Beta and Gamma) captures the banking transactions, the mid-office application (developed by Scrum team Epsilon) processes the transactions and the back-office application (developed by Scrum team Eta) settles the transactions. Scrum team Gamma develops connectivity for the application that is developed by Scrum team Beta. Scrum team Delta develops generic connectivity services. Scrum team Zeta develops applications that support the applications that automate the business process.

At the start of the IAs all Scrum teams record and track the stories in the workflow application. Each team has its own workflow, although *Todo* and *Done* statuses for finished stories are used by each team.

4.2. Performance development

Fig. 1 shows the cycle time development of the feature stories of each Scrum team. On the X-axis the week number is shown. The Y-axis shows the cycle time. Each line represents a Scrum team and each dot the average number of days of the stories that reached the *done* status for that team in that week. A missing dot in a week indicates that no story was closed by the Scrum team in that week. A missing week indicates that no stories were closed by any team. Scrum team Beta and Gamma are combined in one graph because the two Scrum teams use one combined Product backlog.

The first intervention actions started in week 4 (see Fig. 1) and the last intervention actions started in week 18. The first measurement of the total average cycle time was in week 11. The IAs were deployed quite organically, based on the social responses. Some teams needed extensive coaching and direction to keep the pace.

4.3. Intervention results

This section describes the typical responses by the members of the case as a result of the IAs. The typical responses are illustrated by key quotes, collected from the mail application. The labels (in between brackets '[]') are used in Section 6 for reference. The text between '<>' in the quotes is edited text for confidentiality reasons; for instance names of departments or names of team members.

4.3.1. Prioritization

Implementing the priority setting framework and setting matching priority over all Scrum teams started in week 13.

[P1] *"Many thanks for the lively and constructive discussion during the first Product Owner Group (POG) meeting. Below you find the summarized minutes of the meeting. The ultra-short term target for the POG is to understand what has already been developed and drive the development."*, Feature product owner

Weekly Product Owner Group (POG) meetings were planned from week 14 onwards to discuss and match priorities between the Scrum teams. Input to the meeting was the backlog that is high-level prioritized by the Epic product owner. The Scrum team product owners, Feature product owners and the Epic product owner participated in the POG. The Scrum team product owners matched the priority of the team backlogs within and after the POG meeting:

[P2] *"The role of the product owners from each Scrum team is to align the backlogs between the teams. For instance feature X covers <a business function> which requires specific configuration and IT development by each Scrum team."*, Feature product owner communicating the role to others

The prioritization process turned out to be complex and involved many stakeholders. Each stakeholder applied their influence for priority setting towards their interest, which often contradicts with the interest of other stakeholders. Priority needed to be set well prior to the sprint to prioritize refinement activities.

4.3.2. Coordination

Product teams started with standups as of week 18, to share achieved results, next actions and impediments. The realized feature stories were reviewed by the product owners:

[C1] *"I do not understand the flow of events between the applications. It looks like the information is sent between application X and application Y multiple times, while this should be needed once."*, reviewing Feature product owner

A weekly held Scrum of Scrums was institutionalized in week 17–21 to coordinate the work between the Scrum teams. Each team delegated a team member to the Scrum of Scrums which typically was the Scrum Master or a technical lead. The Scum of Scrums allowed the teams to discuss their codependent activities, such as interfacing and integration:

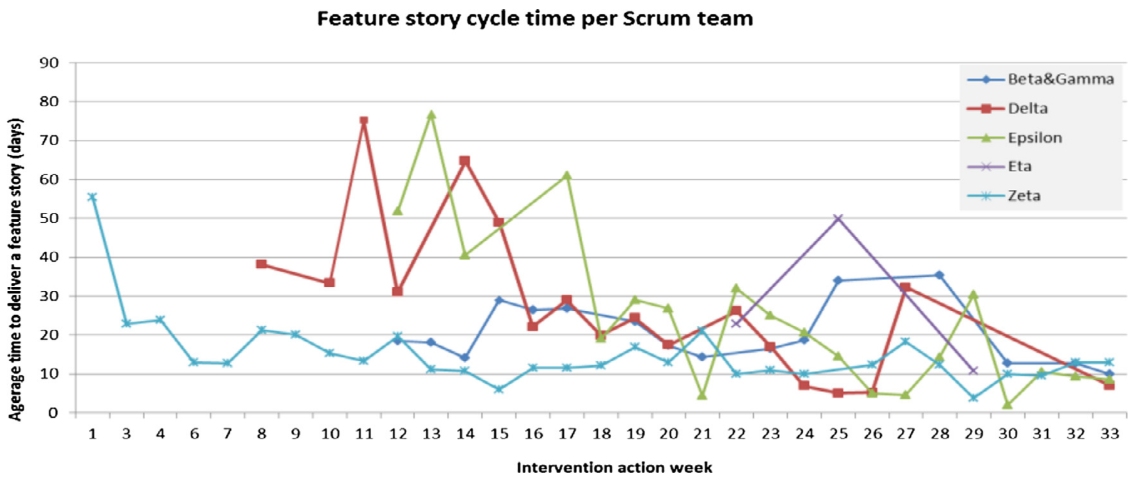

Fig. 1. Front to back business process supported by Scrum teams.

[C2] *"We have a joint view on organizational impediments. We share and leverage best practices across teams and we provide a sounding board from the shop floor….. As far as I know this is the only 'voice from the shop floor', offering direct input to the management team."*, Scrum coach explaining the Scrum of Scrums result

Mini Scrum of Scrums were implemented as of week 12 to support the development of application connectivity between two Scrum teams. Participants of a mini Scrum of Scrums were an interface developer from each of the two Scrum teams and an interface specialist from the generic connectivity Scrum team (Delta). A mini Scrum of Scrums was facilitated by a Scrum Master of the involved Scrum teams.

4.3.3. Alignment

Scrum teams Alpha, Epsilon, Eta and Zeta gradually implemented a four weekly sprint heartbeat, as of week 4. Scrum teams Beta, Gamma and Delta implemented a bi-weekly sprint heartbeat fitting in the four weekly sprint heartbeats.

[M1] *"Thanks for the presentation. One question about the sprints dates. A sprint takes 4 weeks and not one month, which implies that the monthly dates does not fit the 4 week sprint cycle."*, Scrum Master correcting support staff

A single development workflow was implemented in all Scrum teams from week 12 onwards. The workflow was extensively discussed and communicated between stakeholders. An example of such communication is shown in the quote below:

[M2] *"We earlier agreed that the additional workflow testing state is required. Otherwise too many different testing activities are placed under the 'Done' status. The extra status also better aligns with teams that do not develop via the Scrum framework."*, workflow application manager

The workflow was approved by the managers of the Scrum teams in week 13. The workflow was subsequently communicated with the Scrum teams. As a result the teams aligned the test phases between the Scrum teams and mapped the test phases onto the workflow statuses. The Definition of Done (DoD) was discussed and agreed with the Scrum teams. The DoD was integrated with the existing test phases. 'Done' implied that the functionality of a story worked in accordance with the feature stories, including the integration including the application connectivity.

4.3.4. Automation

Linkages between features and stories and the workflow statuses were configured in the workflow application, as of week 4. Several Scrum teams experienced difficulties in correctly connecting the stories to the features in the workflow application, indicated by the missing dots at the left in Fig. 1. Coaching and guidance were required to correct and add the necessary information:

[A1] *"We still miss items in the workflow application., such as (1) Scrum team Beta and Gamma functionality linked to the feature; (2) interface <X> functionality of Scrum team Gamma and (3) Scrum team Alpha functionality for data processing."*, Feature Product Owner

Reporting by the workflow application turned out to be inadequate and Excel was introduced as reporting tool. The Excel report was manually compiled on a weekly basis. The report was compiled with input from the workflow application and Scrum teams. The Excel sheet was subsequently distributed via mail to all stakeholders.

4.3.5. Visibility

The set with IAs was presented during the Product Owner Group (POG) kick-off meeting to the IT managers under which the Scrum teams operate and the product owners in scope of the Scrum teams. The way of working, including roles and responsibilities was afterwards distributed by minutes of meeting.

The workflow application was accessible by all internal employees and each status update by a value chain member was automatically communicated to all members via collaboration tooling. Access to the workflow application was not possible for a supplier that developed software for one of the Scrum teams.

[V2] *"Access is required from <external supplier> to <Scrum team> to be able to have intercompany visibility on the development workflow. This topic was already discussed earlier. It is about providing access to the <workflow application> for external employees. We are still investigating the setup. Technically this is possible."*, workflow application manager

A weekly agenda was distributed to all members of the POG. The agenda included the (1) minutes of last meeting, (2) current status of features and (3) the existing priority of features and (4) the current status of the features and stories in the sprint. The distribution of the agenda triggered the necessary communication between POG members, such as discussing and prioritizing features.

4.4. Focus group results

Scrum Masters, Product Owners and Coordinators in the value chain were each allocated to a focus group. The group with Scrum Masters and the group with Product Owners have 4 members. The Coordinator role coordinates the Scrum team transcending activities in the value chain. That focus group has 5 members.

Each of the groups categorized the items on the post-its. The focus group categorization process was done on a white board by clustering yellow papers while writing and updating the category names.

Table 5
Number of allocated points per category in each team.

Category	Number of allocated points			
Focus group	**One**	**Two**	**Three**	**Total**
Alignment	44 (14%)	9 (3%)	73 (22%)	126 (41%)
Prioritization	27 (8%)	34 (10%)	15 (5%)	76 (23%)
Coordination	5 (2%)	38 (12%)	7 (2%)	50 (15%)
Visibility	3 (1%)	10 (3%)	18 (6%)	31 (10%)
Automation	15 (5%)	9 (3%)	2 (1%)	26 (8%)
Performance	6 (2%)	0 (0%)	10 (3%)	16 (5%)
Total	**100 (31%)**	**100 (31%)**	**125 (38%)**	**325 (100%)**

The items on the post-its confirmed that the performance improvement was achieved with the IAs. Even though the post-it items were independently categorized from the collaboration related issues, the categories were remarkably similar to the IA categories. Table 5 shows the sum of the points per category based on the allocated points per item per focus group member. The table also shows for each focus group and category, the percentage of the total number of allocated points. The total column is the sum of the three focus groups.

Scrum Masters (One) and Product Owners (Three) allocate significantly fewer points to Coordination (2%). Product Owners allocate the largest number of points to Alignment (common sprint heartbeat, workflow, DoR and DoD), while the least number of points are allocated to Alignment by the Coordinators. Focus group Coordinators (Two) allocate the largest number of points to the Coordination IAs. Two focus groups mentioned performance related items, such as: *“Teams are able to pick up more stories”* for which an additional category was created. A few items were referred back to the participant to further explain the item.

The focus groups confirmed the importance of learning during the deployment of the IAs. Typical items on the post-its illustrate that learning process: *“The work between Scrum teams has improved”*, *“Maturity of understanding the (collaboration) process”* and *“Better usage of the workflow tool”*. Learning must be seen as inextricably linked to the deployment of the IAs, and the learning related items were therefore categorized under the other categories.

5. Discussion

5.1. Discussion of the results

In this section the performance development and swift delivery of business value by a set of codependent Scrum teams is discussed and analyzed. The results confirm the effectiveness of the IAs. The cycle time of feature stories in Fig. 1 shows a significant decreasing trend while performing the IAs. The focus groups confirm the effect of the IAs (Vlietland & van Vliet, 2014a, 2015). When referring to the quotes from the focus groups (as included in Section 4), the brackets ‘[]’are used.

The trend in Fig. 1 moves from a total average feature story cycle time of 29 days to 10 days and seems to stabilize at 10 days. A cycle time of 10 days is equivalent to approximately two working weeks, while teams Alpha, Epsilon, Eta and Zeta have a four week sprint cycle [M1]. These results show that stories are delivered faster than the sprint cycle. A driver to deliver faster might be other teams that deliver interdependent stories in a bi-weekly sprint cycle. These teams with a bi-weekly sprint cycle might put social pressure on teams with a four week sprint cycle to deliver faster. Such a premised social factor cannot be validated with the current dataset and is subject for further research.

The prioritization process of a feature affects the feature preparation process and the subsequent sprint cycle [P3]. The quote shows that the preparation process preceding the sprint cycle slows down the feedback loop. For instance, a feature cannot be realized during a sprint due to an unexpected dependency with another feature. In that case the feature priority has to be changed on the backlog. That new prioritized feature has to be prepared prior to the realization in the sprint. The three focus groups confirm the impact of the prioritization IAs (see Table 5), such as slicing work into small sized stories that can be prioritized by a team. To mitigate the longer feedback loop a shorter sprint cycle of 1–2 weeks is suggested.

Coordination by means of the mini Scrum of Scrums resulted in more focus than a Scrum of Scrums, mitigating the disinterest and superficiality in Scrum of Scrums (Paasivaara et al., 2012). The mini Scrum of Scrums stimulated detailing of work prior to the sprint, preventing impediments during the sprint. The focus group with Coordinators allocates 38 points to the Coordination category between Scrum teams. The number of points confirms the need for deeply embedded coordination within and between Scrum teams, confirming the finding of Vlietland and van Vliet (2015).

Entering the data in the workflow application was perceived as difficult [A1], which is confirmed by the jumps between data points in Fig. 1 at the start of the IAs. The reliability of the graph increases over time, even though one of the selection criteria is a single workflow application used by all teams. The combination of visibility, coaching and increased usage of the workflow application stimulated the increase of data entry reliability.

Visibility with the workflow application was limited due to the inaccessibility to external suppliers [V2] and the limitation in reporting had to be mitigated with Excel and email. Further improvements in this area will likely assist in utilizing visibility for swift feedback and mitigating impediments.

The combination of top-down and bottom-up IAs improved the implementation effectiveness. The top-down implementation gave the teams the necessary focus, for instance the prioritization framework [P1]. The bottom-up implementation confirmed the actual adoption, actual commitment and the state of the mental change by the members in the value chain. The bottom-up implementation also utilized feedback about the feasibility of the top-down intervention actions.

The introductory section explained the collaboration related issues that codependent Scrum teams currently face, slowing down the cycle time of new features (Vlietland and van Vliet (2015)). The case study presented in this article shows that a set of IAs can mitigate these issues, resulting in cycle time reduction.

5.2. The Scrum Value chain Framework (SVF)

The IAs are packaged into a Scrum Value chain Framework (SVF) for usage in practice. The IAs are packaged by mapping each IA item onto the framework. The overview of IA items is shown in Table 6. The resulting SVF is shown in Fig. 2.

The blue colored IA items represent the Scrum framework (Sutherland & Schwaber, 2013), the orange colored IA items are additions to that framework, resulting in the SVF. The SVF shows the F2R activities at the left and the I2D activities at the right. At the bottom, the SVF shows the events. Each of the IA items is shown in the SVF. Some of IA items are abbreviated. For instance the IA item: ‘*Epic Product Owner (EPO)*’, is shown as ‘EPO’ with an icon. The ‘Automation (and visibility)’ rectangle at the right summarizes the IA item ‘Workflow Application’.

To validate whether the IA items in the SVF cover all AGF core-elements, each of the IA items is categorized under the five AGF core-elements in Table 6. Each AGF core-element is covered by at least two IA items.

As discussed in Section 5.1 (remark [P3]), the extra ‘Flow to Ready’ phase, that precedes the sprint cycle slows down the feedback cycle. For compensation purposes the sprint duration in the SVF is therefore reduced to 1-2 weeks, instead of 4 weeks as defined in the IAs.

Table 6
Coverage of AGF core elements by IA items.

Role	Event	Team	Artifact	Lifecycle
Feature Product Owner (FPO)	Epic Planning	Product Owner Group (POG)	Feature Description	Flow to Ready (F2R)
Epic Product Owner (EPO)	Bi-daily Feature	Product Team	Aligned Definition of Ready (DoR)	Iterate to Done (I2D)
	Feature Planning	Mini Scrum	Aligned Definition of Done (DoD)	Aligned Sprint Lifecycle
	Scrum of Scrums		Workflow Application	Aligned Sprint Start
	Mini Scrum			
	Feature Review			
	Feature Retrospective			

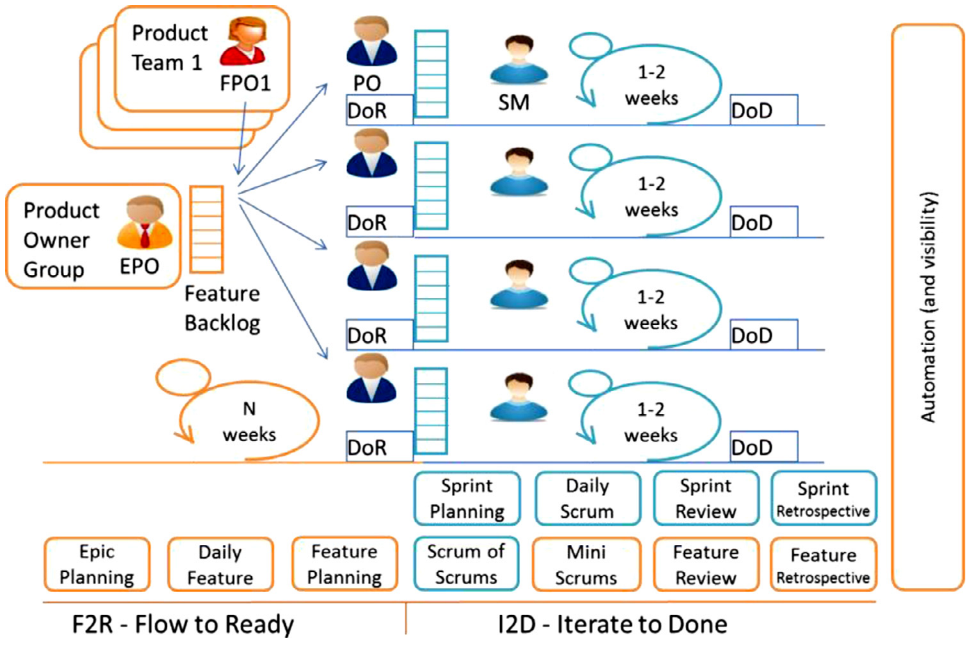

Fig. 2. Scrum Value chain Framework (SVF).

The SVF complies with the Agile manifesto (Beedle et al., 2013) by having a mix of top-down and bottom-up intervention actions. Such a mix is mentioned as a good practice by other authors (Batra, Xia, VanderMeer, & Dutta, 2010; Port & Bui, 2009; Soundararajan & Arthur, 2009). Based on the findings we premise that the SVF offers structure for large scale Scrum as mentioned by Talby and Dubinsky (2009), Soundararajan and Arthur (2009) and Batra et al. (2010), while maintaining the necessary flexibility as intended by the Agile manifesto (Beedle et al., 2013).

6. Threats to validity

For sure, a practical study with IAs in a real-life setting involving multiple teams with real people provides limitations and threats to validity.

First of all, the empirical results come from a single case. Though the IAs were implemented in multiple teams and proved their impact, this is still one case-study in one multinational bank. As such the causal relation between the IAs and the performance improvements cannot be completely generalized. As such, we recommend the repetition of the IAs in more case-studies so as to increase the generalizability for sets of codependent Scrum teams. Secondly, the impact of the combination of the IAs has been validated. The IAs were packaged in the SVF, to be used in organizations that want to decrease the cycle time of their Scrum teams in a codependent setting. Though, each individual IA cannot be traced to the reduction in delivery time, since the actual data was extracted after the IAs were performed, and the effect of an individual IA was not recorded. Another experiment with a different setup is required to determine the effect of each IA independently. Thirdly, the IAs were developed from related work that contains experience reports with similar empirical case studies. As such the external validity of the IAs seems stronger than just a single case. However, the interrelationships between the actions, the level of impact of the individual actions and the balance between them have not been studied in the present research. Furthermore, the IAs were not deployed simultaneously in all teams. Even though the teams were selected based on stability criteria, there might be a bias that also influenced the reduced delivery time. Finally, the relationship between the impact of the IAs and the decreased delivery time with the focus groups was triangulated. As such, there is stronger evidence that the IAs did have an impact in the practical case. Measures were taken to prevent bias in the focus groups, as clarified in Section 3.3.

The SVF needs to be tested in other organizations in the future to provide more evidence. For example, the SVF assumes the Epic product owner to be capable to uniquely prioritize all features. This worked in this empirical case but higher complexity might reduce the decision making effectiveness of the Epic Product Owner. Such decision making effectiveness of the Epic Product Owner requires further study. One might also consider this a generic issue with Scrum by assuming competent role fulfillment. Furthermore, this work has been carried out in a practical setting. Participants in the study, especially the Scrum teams involved, understood that the IAs were taken with a specific purpose. Though, it was not the goal in itself to decrease delivery time specifically, the teams knew that the actions were taken to improve their collaboration, prevent delays and increase the predictability over the value chain. As such, this might have influenced the results (Hawthorne effect). Given the observations and participant opinions in this study, these influences are considered rather limited.

7. Conclusion

In this study a set of practical Intervention Actions (IAs), packaged in a governance framework (SVF), is validated to mitigate collaboration issues in a set of codependent Scrum teams. The IAs resulted in an average delivery time reduction from 29 days to 10 days. The archival records showed the implementation of the IAs, and the delivery metrics confirmed their impact. The participants in the focus groups confirmed the causality between the observed performance improvement and IAs. The results indicate the effectiveness of the IAs and the SVF in a codependent Scrum setting.

The results indicate that the IAs, packaged in the SVF, can help IT service networks realize IT changes faster. Performing IT changes faster enables large companies in the information-intensive industry to swiftly adapt to market changes. Since these companies experience rapid changing business demands, the SVF will likely help companies to achieve a better competitive position, as suggested by (Melville et al., 2004).

Imposing a set of IAs to be interpreted by teams themselves likely introduces new challenges, such as misinterpretations, ignoring a specific action, timely attention to an action, and so on. As such, packaging the results into a single SVF is expected to help mitigate such misinterpretations. Besides recommending to apply the SVF in other settings as to further validate its effectiveness, we recommend repeating the IAs separatelyin other empirical settings too. This is expected to reveal interdependencies between the IAs and the level of impact of the individual IA. A future research avenue therefore is to research the individual IAs, such as qualitatively and quantitatively researching the effect of priority setting onto the delivery time, predictability and efficiency of the set of codependent Scrum teams. Another research avenue is to study prioritization challenges in larger scale settings with multiple feature backlogs and multiple value chains with codependent sets of Scrum teams.

Ideally, individual Scrum teams should cover end-to-end delivery, to prevent the negative impact of dependencies. Looking in perspective at codependent Scrum teams combined with Product Teams, organizations seem to just install another type of waterfall: one of teams instead of development phases. Such a waterfall of teams could never have been the intention of the inventors of Scrum in the first place. However, in complex environments with complex IT landscapes, there is often no real alternative than to start with sets of codependent Scrum teams. In such settings, adopting the IAs and SVF is a best practice to overcome the initial dependencies and to reduce delivery time as soon as possible. Notwithstanding that organizations need to invest in reducing their complexity and enable single Scrum teams to deliver end-to-end value.

References

Akbar, R., Hassan, M.F., Abdullah, A., 2011. A review of prominent work on agile processes software process improvement and process tailoring practices. Software Engineering and Computer Systems. Springer, pp. 571–585.

Ambler, S., 2009. The agile scaling model (ASM): adapting agile methods for complex environments. Retrieved from www.ibm.com website.

Banbury, S., Helman, S., Spearpoint, J., Tremblay, S., 2010. Cracking the bullwhip: team collaboration and performance within a simulated supply chain. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, pp. 1620–1624.

Bartlett, P.A., Julien, D.M., Baines, T.S., 2007. Improving supply chain performance through improved visibility. Int. J. Logist. Manag. 18 (2), 294–313.

Baskerville, R.L., 1999. Investigating information systems with action research. Commun. AIS 2, 2–32.

Baskerville, R.L., Wood-Harper, A.T., 1996. A critical perspective on action research as a method for information systems research. J.Inf. Technol. 11 (3), 235–246.

Batra, D., Xia, W., VanderMeer, D., Dutta, K., 2010. Balancing agile and structured development approaches to successfully manage large distributed software projects: a case study from the cruise line industry. Commun. Assoc. Inf. Syst. 27 (1) (article 21).

Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Highsmith, J.A.H., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Schwaber, K., Sutherland, J., Thomas, D., 2013. Principles behind the Agile Manifesto, from http://agilemanifesto.org/principles.html.

Brown, A.E., Grant, G.G., 2005. Framing the Frameworks: a review of IT governance research. Commun. Assoc. Inf. Syst. 15, 696–712.

Clarke, P., O'Connor, R.V., 2013. An empirical examination of the extent of software process improvement in software SMEs. J. Softw.: Evol. Process 25 (9), 981–998.

Cummings, T., Worley, C., 2014. Organization development and change. Cengage Learning.

Dorairaj, S., Noble, J., Malik, P., 2012. Understanding team dynamics in distributed Agile software development. Agile Processes in Software Engineering and Extreme Programming. Springer, pp. 47–61.

Easterbrook, S., Singer, J., Storey, M.A., Damian, D., 2008. Selecting empirical methods for software engineering research. Guide to Advanced Empirical Software Engineering. Springer, pp. 285–311.

Gersick, C.J.G., 1991. Revolutionary change theories: a multilevel exploration of the punctuated equilibrium paradigm. Acad. Manag. Review 16 (1), 10–36.

Hannan, M.T., Freeman, J., 1984. Structural inertia and organizational change. Am. Sociol. Rev. 49 (2), 149–164.

Holz, H., Melnik, G., 2004. Research on learning software organizations–past, present, and future. Advances in Learning Software Organizations. Springer, pp. 1–6.

Hoyle, D., 2001. ISO 9000: quality systems handbook.

Humble, J., Farley, D., 2010. Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation. Addison-Wesley Professional.

IEEE., 2008. IEEE SA - 829-2008 *Standard for Software and System Test*: The Institute of Electrical and Electronics Engineers.

Ilgen, D.R., Hollenbeck, J.R., Johnson, M., Jundt, D., 2005. Teams in organizations: From input-process-output models to IMOI models. Annu. Rev. Psychol. 56, 517–543.

Jonker, C., van Riemsdijk, M., Vermeulen, B., 2011. Shared mental models. In: Proceedings of the Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems VI, Lecture Notes in Artificial Intelligence, LNAI 6541. Springer Verlag, pp. 132–151.

Jrad, R.B., Ahmed, M.D., Sundaram, D., 2014. Insider action design research a multi-methodological information systems research approach. In: Proceedings of the IEEE 8th International Conference on Research Challenges in Information Science (RCIS). Marrakesh, Marocco, pp. 1–12.

Kniberg, H., Ivarsson, A., 2012. Scaling Agile @ Spotify. Retrieved from https://dl.dropbox.com/u/1018963/Articles/SpotifyScaling.pdf.

Kolb, D.A., 1984. Experiential Learning: Experience as The Source of Learning and Development, Vol. 1. Prentice-Hall, Englewood Cliffs, NJ.

Krueger, R.A., Casey, M., 2008. *A practical guide for applied research*. SAGE Publications, Inc.

Larman, C., Vodde, B., 2013. Scaling Agile Development. CrossTalk 9, 8–12.

Leffingwell, D., 2007. Scaling Software Agility: Best Practices for Large Enterprises. Pearson Education.

Leffingwell, D., 2010. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional.

Lim, B.C., Klein, K.J., 2006. Team mental models and team performance: a field study of the effects of team mental model similarity and accuracy. J. Organ. Behav. 27 (4), 403–418.

Mathieu, J.E., Heffner, T.S., Goodwin, G.F., Salas, E., Cannon-Bowers, J.A., 2000. The influence of shared mental models on team process and performance. J. Appl. Psychol. 85 (2), 273.

Melville, N., Kraemer, K., Gurbaxani, V., 2004. Information Technology and Organizational Performance: An Integrative Model of IT Business Value. Manag. Inf. Syst. Q. 28 (2), 283–322.

Moe, N.B., Dingsoyr, T., Dyba, T., 2008. Understanding self-organizing teams in agile software development. In: Proceedings of the 19th Australian Conference on Software Engineering. Perth, Australia, pp. 76–85.

Moniruzzaman, A., Hossain, D.S.A., 2013. Comparative Study on Agile software development methodologies. arXiv:1307.3356.

Morgan, M., Gibbs, S., Maxwell, K., Britten, N., 2002. Hearing children's voices: methodological issues in conducting focus groups with children aged 7-11 years. Qual. Res. 2 (1), 5–20.

Neely, S., Stolt, S., 2013. Continuous delivery? Easy! just change everything (well, maybe it is not that easy). In: Proceedings of the Agile Conference 2013. Nashville, USA, pp. 121–128.

Olsson, H.H., Alahyari, H., Bosch, J., 2012. Climbing the "stairway to heaven"–a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In: Proceedings of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA). Cesme, Izmir, Turkey, pp. 392–399.

Paasivaara, M., Lassenius, C., Heikkila, V.T., 2012. Inter-team coordination in large-scale globally distributed scrum: do Scrum-of-Scrums really work? In: Proceedings of the 6th IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2012. Lund, Sweden, pp. 235–238.

Pikkarainen, M., Salo, O., Still, J., 2005. Deploying agile practices in organizations: a case study. Software Process Improvement. Springer, pp. 16–27.

Plugge, A., Janssen, M., 2009. Managing change in IT outsourcing arrangements: an offshore service provider perspective on adaptability. Strateg. Outsourcing: Int. J. 2 (3), 257–274.
Port, D., Bui, T., 2009. Simulating mixed agile and plan-based requirements prioritization strategies: proof-of-concept and practical implications. Eur. J. Inf. Syst. 18 (4), 317–331.
Qumer, A., Henderson-Sellers, B., 2008. A framework to support the evaluation, adoption and improvement of agile methods in practice. J. Syst. Softw. 81 (11), 1899–1919.
Rautiainen, K., von Schantz, J., Vahaniitty, J., 2011. Supporting Scaling Agile with Portfolio Management: Case Paf. com. In: Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS). Hawaii, USA, pp. 1–10.
Rising, L., Janoff, N.S., 2000. The Scrum software development process for small teams. IEEE Softw. 17 (4), 26–32.
Romanelli, E., Tushman, M.L., 1994. Organizational transformation as punctuated equilibrium: an empirical test. Acad. Manag. J. 37 (5), 1141–1166.
Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14 (2), 131–164.
Salo, O., Abrahamsson, P., 2005. Integrating agile software development and software process improvement: a longitudinal case study. In: Proceedings of the International Symposium on Empirical Software Engineering (ISESE), 2005. Queensland, Australia, p. 10.
Scheerer, A., Hildenbrand, T., Kude, T., 2014. Coordination in large-scale agile software development: a multiteam systems perspective. In: Proceedings of the 47th Hawaii International Conference on System Science. Hawaii, USA.
Schnitter, J., Mackert, O., 2011. Large-scale agile software development at SAP AG. Evaluation of Novel Approaches to Software Engineering. Springer, pp. 209–220.
Soundararajan, S., Arthur, J.D., 2009. A soft-structured agile framework for larger scale systems development. In: Proceedings of the 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS), pp. 187–195.
Stacey, R.D., 1995. The science of complexity: An alternative perspective for strategic change processes. Strateg. Manag. J. 16 (6), 477–495.
Stelzer, D., Mellis, W., 1998. Success factors of organizational change in software process improvement. Softw. Process: Improv. Pract. 4 (4), 227–250.
Stettina, C.J., Hörz, J., 2015. Agile portfolio management: an empirical perspective on the practice in use. Int. J. Project Manag. 33 (1), 140–152.
Susman, G.I., Evered, R.D., 1978. An assessment of the scientific merits of action research. Adm. Sci. Q. 23, 582–603.
Sutherland, J., 2005. Future of scrum: Parallel pipelining of sprints in complex projects. In: Proceedings of the Agile Conference 2005. Washington, DC, USA, pp. 90–99.
Sutherland, J., Schwaber, K., 2013. The Scrum Guide TM.
Takeuchi, H., Nonaka, I., 1986. The new new product development game. Harv. Bus. Rev. 64 (1), 137–146.
Talby, D., Dubinsky, Y., 2009. Governance of an agile software project. In: Proceedings of the 2009 ICSE Workshop on Software Development Governance. Washington, DC, USA, pp. 40–45.
Team, C.P., 2010. CMMI for Development, version 1.3.
TFSC., 2011. Retrieved from the changing face of payments - a review of current payments infrastructures, drivers for change and implications for the future.
Unterkalmsteiner, M., Gorschek, T., Islam, A.M., Cheng, C.K., Permadi, R.B., Feldt, R., 2012. Evaluation and measurement of software process improvement—A systematic literature review. , IEEE Trans. Softw. Eng. 38 (2), 398–424.
Vacanti, D., Vallet, B., 2014. Actionable Metrics at Siemens Health Services.
van Bon, J., Jong, A., Kolthof, A., 2007. Foundations of IT Service Management Based on ITIL, Vol. 3. Van Haren Publishing.
Van Tiem, D.M., Karve, S., Rosenzweig, J., 2006. Hidden order of human performance technology. In: Handbook of Human Performance Technology, 1251. Cyprus. Pfeiffer, pp. 1251–1273.
VersionOne, 2013. Retrieved from 7th Annual State of Agile Development Survey.
Vlaanderen, K., Jansen, S., Brinkkemper, S., Jaspers, E., 2011. The agile requirements refinery: applying SCRUM principles to software product management. Inf. Softw. Technol. 53 (1), 58–70.
Vlietland, J., van Vliet, H., 2014. Alignment issues in chains of Scrum teams. In: Proceedings of the 5th International Conference on Software Business. Cyprus, pp. 301–306.
Vlietland, J., van Vliet, H., 2014. Improving IT incident handling performance with information visibility. J. Softw.: Evol. Process 2014 (26), 1106–1127. doi:10.1002/smr.1649.
Vlietland, J., van Vliet, H., 2015. Towards a governance framework for chains of Scrum teams. J. Inf. Softw. Technol. 57, 52–65. doi:10.1016/j.infsof.2014.08.008.
Walsham, G., 1995. The emergence of interpretivism in IS research. Inf. Syst. Res. 6 (4), 376–394.
Weick, K., Quinn, R., 1999. Organizational change and development. Annu. Rev. Psychol. 50 (1), 361–386.
Yamakami, T., 2013. Self-innovation skill-based change management: an approach toward flexible organizational management. In: Proceedings of the 3rd International Conference on Cloud and Green Computing, pp. 256–260.

Jan Vlietland is a managing director at Search4Solutions. His experience centers on directing large improvement programs in information-intensive industries. He holds a Master of Science in Management from the Open University in the Netherlands. He got a PhD from the Department of Computer Science, VU University, Amsterdam, in 2015.

Rini van Solingen is a part-time full professor in Global Software Engineering at the Delft University of Technology in The Netherlands since 2010. In addition, he is the chief technology officer at Prowareness, an IT consultancy and offshore company in the Netherlands. He received his master of science in Technical Informatics from Delft University in Technology and holds a Ph.D. in Technology Management from the Eindhoven University of Technology, both in The Netherlands. Rini is author of The Power of Scrum, and Scrum for Managers.

Hans van Vliet is a Professor in Software Engineering at the VU University Amsterdam, The Netherlands, since 1986. He got his PhD from the University of Amsterdam. His research interests include software architecture, knowledge management in software development, global software development, and empirical software engineering. Before joining the VU University, he worked as a researcher at the Centrum voor Wiskunde en Informatica (CWI, Amsterdam). He spent a year as a visiting researcher at the IBM Almaden Research Center in San Jose, California. He is the author of "Software Engineering: Principles and Practice", published by Wiley (3rd Edition, 2008). He is a member of IFIP Working Group 2.10 on software architecture, and the Editor in Chief of the Journal of Systems and Software.