VU VRIJE UNIVERSITEIT AMSTERDAM

VU Research Portal

Improving the agility of IT service networks

Vlietland, J.J.

2015

document version
Publisher's PDF, also known as Version of record

Link to publication in VU Research Portal

citation for published version (APA)
Vlietland, J. J. (2015). *Improving the agility of IT service networks*. [PhD-Thesis - Research and graduation internal, Vrije Universiteit Amsterdam].

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:
vuresearchportal.ub@vu.nl

Download date: 13. Mar. 2024

Chapter 4

Towards a governance framework for chains of Scrum teams

Context: Large companies operating in the information intensive industries increasingly adopt Agile/Scrum to swiftly change IT functionality because of rapid changing business demands. IT functionality in large enterprises however is typically delivered by a portfolio of interdependent software applications involving a chain of Scrum teams. Usually, each application from the portfolio is allocated to a single Scrum team, which necessitates collaboration between the Scrum teams to jointly deliver functionality.

Objective: Identify the collaboration related issues in chains of Scrum teams.

Method: We used a qualitative approach with transcripted interviews from three case studies that were coded and analyzed to identify the issues.

Results: We identified six issues in chains of codependent Scrum teams; coordination, prioritization, alignment, automation, predictability and visibility. The synthesis of these issues with existing theory resulted in nine propositions. These nine propositions have been combined into a conceptual model.

Conclusion: We propose this conceptual model as a starting point for a governance framework to manage chains of Scrum teams that addresses the identified issues.

4.1 Introduction

Large companies operating in the information intensive industries experience rapid changing business demands that require swift delivery of new IT functionality. To be able to deliver such IT functionality swiftly internal IT development centers increasingly adopt Agile methods. A common Agile method is Scrum which aims to empower IT development centers to deliver customer focused IT functionality in a fast pace.

IT functionality in large companies however is delivered by a portfolio of interdependent applications, not just a single application. Each application in the portfolio supports a business function in the front to back business process. Typical front to back business functions are: front-office, mid-office, back-office and finance. Figure 21 illustrates a typical front to back business process with business functions for the mortgage business line.

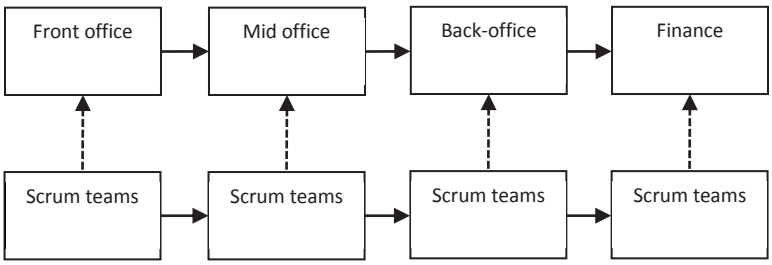

Figure 21, Front to back business process supported by Scrum teams

The front-office – in this example – performs customer facing processes, such as a mortgage client contact center. The mid-office calculates risk by a credibility check of a new client. The back-office performs the mortgage and settlement process, such as account opening. The finance function takes care of the actual funds provisioning, typically performed for multiple business lines.

Scrum is an Agile based method for incremental software development that uses low boundary cross-functional collaboration in software development teams that work toward a set team goal (Schwaber, 2004, 2011). 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. Scrum defines three roles: the Product Owner, Scrum Master and other Scrum team members. 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. A Scrum team has a small size (max 10).

The small team size eases intra-team knowledge sharing and utilizing the self-organizing ability in professional teams (Takeuchi & Nonaka, 1986). These self-organizing practices are encouraged by a structure, containing a product backlog, sprint backlog, sprint planning, daily-standups and a sprint review (Moe, Dingsoyr, & Dyba, 2008). The team has the task to develop software based on the sprint backlog (Rising & Janoff, 2000).

Scrum teams can be mapped in different ways onto the application landscape. Some prefer to have one Scrum team for the whole front to back chain. However two constraints make such front to back coverage difficult. First, the amount of involved IT staff then easily exceeds the generally agreed upon maximum Scrum team size of 10 members. Second, changes require highly specialized skills that cannot be shared easily. These two constraints result in dedicated Scrum teams for each business function in the front to back process, as shown in Figure 21. Each of these dedicated Scrum team delivers application functions that merged together results in features that automates the front to back business process. We define features as: 'intentional distinguishing characteristics of the application landscape that can be used by a business user', such as the mortgage registration feature.

Given the interdependencies in the application chain, multiple Scrum teams then need to jointly deliver new or changed features, as shown in Figure 21. Joint delivery implies that Scrum teams need to collaborate. Particularly the high frequency of deliveries which are common in Scrum settings likely makes efficient collaboration an important performance factor (Dorairaj, Noble, & Malik, 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 10 members, multidisciplinary team, typically owned IT applications, high-frequency deliveries and focus on a single backlog. The focus on the single backlog in combination with the 'owned' IT applications likely results in a bounded Scrum team focus rather than a feature delivery focus. Such focus likely results in collaboration (related) issues.

The application of Agile methods in large organizations has been subject of research for more than ten years (Cockburn, 2006; Dingsøyr, Itkonen, & Fægri, 2013). The majority of reported results are experience reports and there is a need for developing theory that answers burning practitioner questions (Dingsøyr & Moe, 2013; Freudenberg & Sharp, 2010; Jalali & Wohlin, 2012; Martini, Pareto, & Bosch, 2013b).
In this study we empirically identify the issues in chains of Scrum teams and develop new theory. We used a qualitative approach with transcripted interviews from three case studies, that were coded and analyzed to identify the issues. We identified six issues: (1) a lack of coordination in the chain (2) mismatches in backlog priority between teams, (3) alignment issues between teams, (4) a lack of IT chain process

automation, (5) unpredictability of delivery to commitment and (6) a lack of information visibility in the chain. We subsequently build a first version of a conceptual model with these six issues. This conceptual model can serve as a starting point for the development of a governance framework to mitigate the identified issues in chains of Scrum teams. The chapter is an extended version of Vlietland and van Vliet (2014a), in which we presented a preliminary analysis of two case studies, and identified four issues. The present chapter contains a more elaborate discussion of these two cases as well as an additional case, resulting in a larger and refined set of issues. Also, the synthesis of these issues into a conceptual model is new material. Validation of the conceptual model is a topic for future research.

The remainder of this chapter is organized as follows. Section 4.2 covers related work. Section 4.3 explains the research method. Section 4.4 elaborates on the case studies and the issues identified. Section 4.5 synthesizes these findings into a conceptual model. Section 4.6 elaborates on the threats to validity. Section 4.7 concludes the study, deduces implications and suggests future research avenues.

4.2 Related Work

As we did not find literature addressing chains of Scrum teams, we expanded our literature study to Agile related issues in the enterprise. Our literature study acknowledged three categories of related work: scaling issues, 3C issues and automation issues. We identified multiple issues in each category.

Table 14 shows the overview of the identified issues in each category of related work. In general, different publications identify different issues, as explained in more detail in subsection 4.2.1 to 4.2.3. Since large body of literature offers solutions for Agile related issues, we also discuss the solution based literature in this section.

Table 14, Overview of identified issues in each related work category

Category	Identified issues / solutions
Scaling issues	Priority mismatches (3x), Test effort and coverage, Increased management overhead, Increased configuration management effort, Communication issues, Lack of information, Requirements gathering problems, Limited business feedback, Dependencies between Definitions of Done.
3C issues	Soft-standardizing managing work (2x), Operational visibility and transparency (2x), Unclear requirements and design, Reduced informal contact, Inconsistent work practices, Coordination dependencies, Scrum of Scrums
Automation issues	Sensitivity of integration mistakes blocking integration automation, Traditional tooling making development unnecessary complex, Heterogeneous configurations, Lack of transparency and a short feedback cycle, Challenging cultural shift to new values, Struggle with traditional processes hindering 3C and continuous improvements, Team specific test environments being incompatible with the integrated test environment.

A part of the related work concerns distributed contexts. Ågerfalk et al. (2005) identified three main issues arising with the distribution of software development practices: (1) spatial separation, (2) time-zone differences and (3) cultural differences. Spatial separation leads to exacerbation of communication and coordination (Ågerfalk et al., 2005). As communication and coordination is related to collaboration we include related work about distributed contexts in our literature study. We exclude the cultural and time-zone issues (Woodward, Surdek, & Ganis, 2010), since our study environment has a shared culture and time-zone.

Given the Scrum specific characteristics we narrow the literature study to Scrum setups. Sutherland et al. (2007)consider three models for collaborating Scrum teams in a distributed context: (1) Isolated Scrums – teams are geographically isolated, (2) Distributed Scrum of Scrums – teams are geographically isolated and integrated by Scrum of Scrums and (3) Totally integrated Scrums – Scrum teams are cross-functional with members distributed across geographies. Our literature study excludes the latter, given the characteristics of our study.

The remainder of this section is organized as follows: subsection 4.2.1 discusses related work about Agile scaling issues, subsection 4.2.2 discusses specific collaboration, coordination and communication related issues and subsection 4.2.3 discusses issues with regard to the automation of the software development processes.

4.2.1 Scaling issues when applying Scrum in large enterprises

The large enterprise with collections of Scrum teams is far more complex than a single Scrum team and faces many scaling issues (Ambler, 2012). Some researchers studied these Agile scaling issues when adopting Scrum in the enterprise. These scaled Agile studies identified a diverse set of issues, except for priority (related) issues which are mentioned by several studies.

Petersen and Wohlin (2009) studied a case of Agile projects with Scrum characteristics, and identified issues with (1) creating and maintaining the priority list, (2) effort to setup and maintain a test basis that covers a sufficient part of the applications, (3) increased management overhead due to a high number of teams that requires coordination and communication and (4) increased configuration management effort due to an increased number of releases. The paper does not indicate whether the issues are identified in a Scrum chain setting.

The issue with prioritization has also been identified by Lehto and Rautiainen (2009), who studied Scrum issues in a mid-size software company, while the setting of the

Scrum teams is ambiguous. In their study they conclude that prioritization of the high-level goals was unclear which impedes organizing and tracking development work. They also identified traceability issues from high level goals to detailed plans and a lack of information about progress which made it impossible to take corrective actions in time. Waardenburg and van Vliet (2012) studied two enterprises with Scrum teams. While the paper does not elucidate whether the teams are part of a chain, they also identified issues with prioritization, as a consequence of the lack of business involvement. Other identified issues are (1) requirements gathering problems, (2) limited business feedback, (3) communication problems and (4) dependencies between Definitions of Done.

Other related work report experience with scaling. Saddington (2012) reports a success case about scaled product ownership in multiple codependent Scrum teams. They state that visibility and alignment of vision, goals, teams and workload were essential elements to the success of the project. Rautiainen et al. (2011) describe a company that implements an Agile portfolio management structure to solve prioritization issues. Listing the projects in priority order on a single backlog creates visibility about ongoing projects, which benefits the coordination between Scrum teams.

Even though Agile principles aim to introduce flexibility the need for plans and structure remains (Talby & Dubinsky, 2009). Batra, Xia, VanderMeer, and Dutta (2010) conducted a case study on a large project and conclude that combining Agility with traditional plan-driven methods is essential for having both flexibility and control. Soundararajan and Arthur (2009) argue that Agile practices need to be structured to develop large software systems. Their proposed soft structured framework consists of a requirements gathering approach and a tailored development process. A comprehensive structure for a scaled Agile application in the enterprise is the Scaled Agile Framework (SAFe) of. The framework targets seven areas to achieve parallel Scrum development: (1) cross-functional teams, (2) standardized planning and tracking, (3) standardized iterations, (4) smaller, frequent releases, (5) concurrent testing, (6) continuous integration and (7) regular reflection and adaptation.

4.2.2 3C issues when applying Scrum in large enterprises

Codependent chains of Scrum teams (see Figure 21) develop features in parallel (Sutherland, 2005), which requires collaboration, coordination and communication (3C) between the Scrum teams (Sharp & Robinson, 2010). In this subsection we discuss Agile related work that identifies 3C issues.

We follow existing literature to define 3C. Collaboration is: "*the process of two or more people working together on a task*" (Henneman, Lee, & Cohen, 1995; Sharp &

Robinson, 2010). Communication is: "*the exchange of information or knowledge through verbal or non-verbal means between two or more people*" and coordination is: "the *process of managing dependencies among activities*" (Calvert, 1995; DeSanctis & Jackson, 1994; Sahin & Robinson, 2005).

Oppenheim et al. (2011) studied two Agile cases of collaboration between enterprises and present an architecture for such collaboration. They identified three challenges: (1) a standard way to manage similar work, yet allowing local variations, (2) overall operational visibility and transparency and (3) clarity about requirements and design. The need for visibility is also identified by Hildenbrand et al. (2008) which systematically analyzed the distributed software development scenario. They conclude that the distribution of teams hinders collaboration due to the virtual environment that limits body gestures and facial expressions. These limitations might also exist in chains of Scrum teams as the teams are more or less isolated and rely on virtual environments. (S. Lee & Yong, 2010) studied a combination of a global product team and three local IT teams and highlighted successful practices and challenges. They identified three main issues (1) a lack of planning and communication between teams, (2) a combination of low priority and inexperienced staff and (3) mixed responsibilities. Sharp and Robinson (2008) studied the operational collaboration of three Agile teams in different companies. They conclude that even though teams work in the same time-zone they face the problem of maintaining an informal but disciplined collaboration and coordination structure. Such need for structure is supported by the scaled Agile literature (see subsection 4.2.1). Collaboration between Scrum teams seems to be limited by design, because a Scrum team's goal is to realize its own backlog items. Schwaber (2011) supports this view by mentioning the unlikeliness of Scrum teams to collaborate and discusses a case of using product integration teams for spanning team boundaries.

We earlier referred to Sutherland et al. (2007), who consider three models for collaborating Scrum teams in a distributed context. Reported research on such collaborations (Hildenbrand et al., 2008; Oppenheim et al., 2011; Sharp & Robinson, 2010) do not unambiguously identify the model they consider. Our experience from practice is that, generally, aspects of the 'Distributed Scrum of Scrums' and 'Totally integrated Scrums' models apply.

Codependencies between Scrum teams seems to need coordination (Larman & Vodde, 2013). Coordination between Scrum teams is typically done by Scrum of Scrum meetings in which Scrum Masters from all teams participate. However such a Scrum of Scrums can be problematic. Paasivaara et al. (2012) conducted a multiple case study on how a Scrum of Scrum is applied, and concluded that they seem to work poorly in case of too many participants (15-20) and disjoint interests and concerns. A way to

enable coordination between Scrum teams is to implement more elaborate structures, such as mentioned by Schnitter and Mackert (2011). They define product teams for coordinating work over multiple Scrum teams. These product teams consist of a product manager, product team Scrum Master, software architect, delivery manager, knowledge management & documentation expert, UI designer and a stakeholder representative. For managing more than seven Scrum teams an intermediate organizational layer is suggested between the product teams and Scrum teams to cater for the necessary coordination (Schnitter & Mackert, 2011). We found no other studies about coordination in chains of Scrum teams (Sutherland et al., 2008).

Yet, other related work is found in the area of global software development that report issues or experiences between groups. Bannerman, Hossain, and Jeffery (2012) identified four types of coordination challenges from global software development literature: (1) reduced informal contact leading to a lack of task awareness, (2) inconsistent work practices that impinge coordination. The two other items are related to time-zone and socio-cultural distance and are therefore not in our scope. Begel et al. (2009) surveyed 775 software engineers at a multinational software company and identified artifacts for coordination between development teams. They identified the artifacts release schedule, features, API's, bugs, documentation, code, prioritized work items and status, as the top coordination dependencies. Schedules and features are considered the common objects to coordinate work. Strode, Huff, Hope, and Link (2012) provide a model to coordinate Agile distributed projects. The model is based on three principles: (1) synchronization of activities, (2) an inter-team structure and (3) boundary spanning activities and artifacts.

Related work also exists in the area of communication, though the work does not specifically research communication issues in Scrum chains. Martini, Pareto, and Bosch (2013a) investigate, through a survey, communication factors affecting both speed and reuse in 3 large companies, identifying five Agile team interfaces: (1) system engineers, (2) product management, (3) distributed teams, (4) inter-project teams and (5) sales unit.

Green, Mazzuchi, and Sarkani (2010) studied the need for communication in the development phases of a distributed project. They conclude that in a distributed Agile setting communication is particularly indispensable during the beginning of the development project. The argumentation is that the beginning of the project is most turbulent and therefore needs most communication to understand (changing) requirements.

Building a shared mental model might be a more fundamental reason for the need for communication in that phase (Mathieu, Heffner, Goodwin, Salas, & Cannon-Bowers,

2000), as such a shared mental model benefits the interpretation of shared information.

Mishra and Mishra (2011) studied a complex development project with multiple teams. The large project size brought the necessity of (1) having sufficient communication between Scrum teams to synchronize work, (2) informing each other about progress and (3) discussing unresolved issues leading to adjusted plans. They mention brainstorming and iteration planning meetings as instruments to share information. Another way to achieve communication is using regular cross-team meetings for informing each other about progress and issues. They conclude that (1) the large size of the project required certain levels of control and (2) flexibility and architectural designs helped to communicate a clearer picture of the entire system. Most of their implications for practice concern business-IT alignment, not communication between Scrum teams. A practical way to organize cross team communication is presented by Kniberg and Ivarsson (2012). Their article reports about the use of tribes at Spotify to cater for the necessary communication. They describe an organizational setup for multiple teams to stimulate knowledge sharing and communication between the teams.

4.2.3 Automation issues when applying Scrum in large enterprises

One of the key successes of information technology (IT) is the automation of business processes. IT can much quicker execute repeated tasks and process vast amounts of data. The same principle is now increasingly applied to software engineering processes. Automation of software engineering processes enables very short time to market of new software, such as at Facebook (Feitelson et al., 2013). The Agile community underlines the importance by making software engineering process automation one of their core Agile principles (Beedle et al., 2013). A. W. Brown, Ambler, and Royce (2013) used a software cost perspective to explain the benefits of automation. They simplified a typical software cost model with dozens of parameters to three essential parameters. They conclude that software process automation is one of the essential parameters to reduce the time to market of software releases.

The automation of software engineering processes concerns the automating of the integration, testing and deployment processes. The key characteristics are Continuous Integration, Continuous Testing and Continuous Deployment, part of the umbrella Continuous Delivery (Beedle et al., 2013; Humble & Farley, 2010).

Continuous Integration stands for the immediate build and integration of a new piece of software as soon as it is ready and checked into the configuration management system (Garg, 2009; Humble & Farley, 2010; Jyothi & Rao, 2011; Ståhl & Bosch, 2013,

2014). Kim, Park, Yun, and Lee (2008) briefly mention two Continuous Integration related issues in their paper: (1) the sensitivity of integration mistakes by the packaging maintainer which blocks the integration automation and (2) developers using own testing environments which are incompatible with the integrated testing environment, blocking the integration test.

Continuous Testing is synonym for the automated execution of a series of automatic tests, verifying the integrity of the updated system (Candea, Bucur, & Zamfir, 2010; Humble & Farley, 2010; Muslu, Brun, & Meliou, 2013). The papers about Continuous Testing that we found during the literature study, such as the experience papers of Muslu et al. (2013) and Candea et al. (2010), do not mention any Continuous Testing related issues.

Continuous Deployment is the discipline of automated distribution of the new version of the system from the development stage to the production stage (Feitelson et al., 2013; Hardion et al., 2013; Humble & Farley, 2010). Olsson et al. (2012) present a multiple-case study in which key-barriers associated with the transition towards Continuous Deployment are identified, being (1) the struggle with traditional processes that hindered 3C and continuous improvements, (2) traditional tooling that made development unnecessary complex and heterogeneous configurations, (3) a challenging cultural shift to new values and (4) a lack of transparency and a short feedback cycle.

Recently, Fitzgerald and Stol (2014) stretched the 'Continuous' concept and practice with the introduction of 'Continuous *'. The Continuous * concept includes Continuous Planning, which involves dynamic open-ended plans that evolve in response to changes in the business environment, and thus involve a tighter integration between planning and execution (Fitzgerald & Stol, 2014; Knight, Rabideau, Chien, Engelhardt, & Sherwood, 2001). The Continuous Planning concept involves planning and tracking of the software delivery process, while Continuous Delivery comprises the automation of the software manufacturing and software delivery itself. We found no papers that mention any Continuous Planning related issues.

We conclude that most of the existing related work about software engineering automation concerns case studies about automation technology. Issues in software engineering process automation are scarcely mentioned and we found no studies about automation issues in chains of Scrum teams.

4.3 Research Method

As existing academic literature provides insufficient answers to our research question, we selected and performed (Saunders et al., 2009) a holistic multiple-case study at multinational service providers, to gain an in-depth understanding of the issues in interdependent chains of Scrum teams and develop new theory (Dul & Hak, 2012). A method for inductive research that builds new theory is presented by Eisenhardt (1989). Her work syntheses previous work on qualitative methods, case study research and grounded theory building (Glaser & Strauss, 1967; Huberman & Miles, 1984; Yin 1981). She presents a process of building theory from case study research that has the steps: (1) getting started, (2) selecting cases, (3) crafting instruments and protocols, (4) entering the field, (5) analyzing data, (6) shaping hypotheses, (7) enfolding literature and (8) reaching closure.

We use Eisenhardt (1989) for our research method. We slightly adapted the method to fit our needs by splitting the design into two sequential stages as shown in Figure 22. Stage A targets Scrum chain selection and role mapping, which prepares stage B that aims finding the issues.

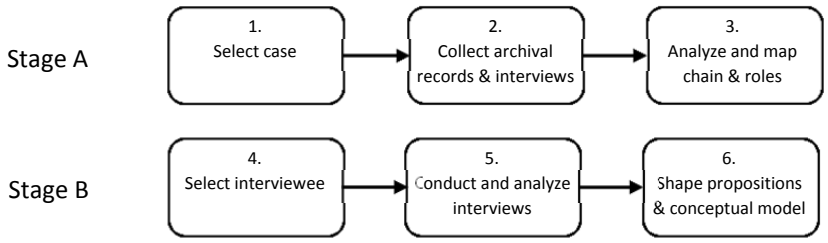

Figure 22, Research design

The research starts with selecting an interdependent chain of Scrum teams by using selection criteria as shown in Table 15. We follow the project level contextualization attributes of Kruchten (2013) to develop the criteria for our study. The attributes of Kruchten help to identify the context of an Agile environment. The attributes are size, architecture stability, business model, team distribution, change rate, system age, criticality and governance. The selection criteria enable the identification of the unique characteristics of a chain of Scrum teams and enhance the content validity of the research.

Table 15, Case study selection criteria

Selection category	Selection criteria for each team in the chain
Application interdependencies	Applications have interdependencies with other applications in the front to back chain.
Chain setup	Scrum teams are part of a chain and each application under development is allocated to one Scrum team.
Team distribution	Studied Scrum team members are working in the same country.
Culture	Studied Scrum team members have the same nationality.
Scrum roles	Scrum roles product owner, Scrum Master and Scrum team are assigned.
Scrum processes	Scrum sprints, sprint planning meeting, daily standups, sprint demo and sprint retrospectives are performed
Scrum artifacts	Product backlog and Scrum backlog is available and product increments are delivered

The chain of Scrum teams is selected in step 2, by collecting archival records and conducting interviews by phone. These interviews are annotated in an interview log. The data is subsequently analyzed to map the chain and identify the involved roles. Steps 2 and 3 iterate until the mapping is complete. Subject matter experts working in the network are involved to secure the validity of the mapped results (Gibbert & Ruigrok, 2010).

The interviewees are selected in step 4, based on the identified roles in step 3. The interview process is flexible; in case another involved role is identified during the role can be included in the interviews.

The interviews itself are conducted in step 5. Open-ended questions are prepared to guide the interviews, as shown in Table 16. We use Galbraith's (1995) framework to study a broad range of aspects in the chain of Scrum teams. Galbraith's framework has five dimensions to assess organizations: (1) strategy, which determines direction, (2) structure, which determines the location of decision-making power, (3) processes have to do with the flow of information; they are the means of responding to information technologies, (4) rewards provide motivation and incentives for desired behavior and (5) people, which is about the selection and development of the right people.

Each of Galbraith's dimensions is translated to our research context. We use the (1) strategy dimension to find the issues in translation of strategy to operational plans and the (2) structure dimension to understand and find the issues in the existing roles and responsibilities. We use Galbraith's (5) people dimension to assess the issues in

mindset and competences. Galbraith's (4) reward dimension is kept out of the scope of this research, as analyzing the issues in human resource performance management does not fit in this study. We use Galbraith's (3) process dimension to find process related issues between Scrum teams. Galbraith's process dimension is specified in terms of collaboration, coordination and communication (see Table 16), which have a key role in Agile settings as discussed in subsection 4.2.2.

Table 16, Predefined interview questions

Question	Grounding
What are the main strategic to operational level translation issues with the Scrum chain setup?	Galbraith's strategy
What are the main roles and responsibilities related issues with Scrum chain setup?	Galbraith's structure
What are the main inter Scrum team coordination issues with the Scrum chain setup?	Galbraith's process, 3C coordination
What are the main inter Scrum team collaboration issues with the Scrum chain setup?	Galbraith's process, 3C collaboration
What are the main inter Scrum team communication issues with the Scrum chain setup?	Galbraith's process, 3C communication
What are the main mind-set and competence related issues with the Scrum chain setup?	Galbraith's people

Step 5 starts with one interview that is subsequently analyzed. Follow-up interviews are used to refine the script and the analyzed results. Such iterative approach allows script optimization for successive interviews and the mitigation of potential problems as summarized by Myers and Newman (2007). The interviews are digitally recorded for content- and construct validity purposes (Mentzer & Flint, 1997).

The interview setup is built on the dramaturgical model, using the metaphor of a theatre (Myers & Newman, 2007). The dramaturgical model is based on the general theory of Goffman (1959) that sees social interactions as a drama, with actors that perform in a variety of settings using a script that guides behavior. During the interview a delicate balance is kept between providing direction and getting unbiased answers, while mitigating the potential interviewing pitfalls as summarized by Myers and Newman (2007). The objective and the topics of the interview are set at the beginning of the interview, to set a framework that can be gently referred to in case the interview moves off-topic.

The interview results are transcripted and qualitative analysis techniques are used to analyze the transcripted data (Dul & Hak, 2012; Yin 2009). The qualitative analysis starts by identifying quotes in the transcriptions (Saunders et al., 2009). The quotes are tagged with open codes. The open coding process is performed in a number of iterations (Saldaña, 2012). If a quote applies to multiple open codes the quote is tagged with all applicable codes. For instance the quote: '*The application that is developed by the team is connected with an interdependent application via an enterprise service bus",* is tagged with the open codes *'ApplicationInterdependencies'* and *'ApplicationDevelopment'*.

Open coding proceeds line-by-line, until patterns emerge. These patterns of open codes are grouped into categories, the axial coding process. The axial coding process is done in two ways. First, codes are grouped by prefixes in the tags. For instance all codes which are related to issues are tagged by the prefix '*Issue*'. Second, codes are grouped in code families. For instance performance metric related codes are placed in the code family '*PerformanceMetrics*'.

Data collection continues until a new transcription does not significantly contribute to knowledge and insight (Dul & Hak, 2012). Significantly contributing is quantified by determining whether an additional interview results in more than 5% new or modified codes. In case an interview results in more than 5% new or modified codes an additional interview is conducted. This approach is in line with Sandelowski (1995) and Marshall (1996).

After the axial coding process the open codes tagged with five or more quotes, are clustered into the main categories for which supercodes are used. Supercodes are queries for retrieving a selected set of codes. For instance a query can be built to retrieve all quotes that are linked to open codes that have the word '*Issue*' in their name at one company. The supercodes are used to ground the concepts, which are the (key) issues. All steps of the data analysis are recorded in Atlas TI, a CAQDAS package (Gibbert & Ruigrok, 2010; Saunders et al., 2009).

The issues are used during step 6 to define the propositions. The relationships between the issues that result in the propositions are grounded in existing theory by an additional study of related work. The propositions are then used to build the conceptual model (Birks & Mills, 2011).

4.4 Issues identified

Using our professional network, we identified five cases at large multi-national organizations on which we applied the selection criteria. Three cases passed the criteria and were subsequently used in our research. We performed one case study at a telecommunication company (9 interviews in two overlapping chains), one at a retail bank (6 interviews in one chain) and one at an insurance company (3 interviews in one chain). Each of the companies has a centralized IT organization with 250-1500 IT development employees who offer application services to front to back business functions.

For the interviews, we selected the key roles Product Owners, line managers and Scrum Masters. We noticed in the preliminary interviews that the role of Scrum Masters was overlapping with that of the team members, in that the Scrum masters also performed development work. Also the Scrum literature considers Scrum Masters part of the development team. We therefore decided that interviewing the Scrum Master was sufficient and no other Scrum team members were interviewed. We transcribed the interviews and coded them as described in Section 4.3. Table 17 shows for each identified issue a brief description and the grounding of each issue by the number of quotes.

Table 17, Code grounding of each issue

Issue description	Issue	Nr of quotes
A lack of coordination in the chain	Coordination	124
Mismatches in backlog priority between teams	Prioritization	122
Alignment issues between teams	Alignment	99
A lack of IT chain process automation,	Automation	72
Unpredictability of delivery to commitment	Predictability	56
A lack of information visibility in the chain.	Visibility	38
	Total	511

Table 18 shows for each issue the percentage of quotes in each case study. The table shows that most of the coordination issues occur in case 1. Most prioritization and alignment issues occur in case 2. The highest percentage of predictability and visibility issues occurs in case 3. Automation issues are (relatively) equally often mentioned in all three cases.

Subsections 4.4.1 – 4.4.3 contain, for each case, key quotes for each concept. The issues in each subsection are sorted in the order in which they appear in Table 18. The quotes are labeled between the brackets '[ ]'. These labels are used in section 5 to refer back to the quotes

Table 18, Cross-case analysis of the issues

Issue	Case 1	Case 2	Case 3
Coordination	29%	16%	18%
Prioritization	22%	35%	17%
Alignment	20%	22%	13%
Automation	14%	15%	15%
Predictability	11%	5%	17%
Visibility	4%	7%	20%
Total	100%	100%	100%

4.4.1 Case study 1: Retail banking front to back application chain

The retail bank has a centralized IT organization with 150 Scrum teams that deliver web application services to the various business lines that deliver the services to the banking customer. The web applications are developed by front-end Scrum teams that are clustered around different channels (e.g. internet, call center). The web applications interact with back-office applications that are clustered around financial products. The back-office applications are mainly commercial of the shelf (COTS) packages. These packages are configured by Scrum teams. The back-office packages interact with finance applications that are developed by waterfall organized development teams. The organization has allocated an integrator role, next to the regular Scrum-roles. The integrator is responsible for, monitors and coordinates front to back feature development.

Coordination: Issues regarding coordination typically express themselves during testing and deployment, because in the end of the development lifecycle all teams should have delivered their piece of the puzzle. To ensure that the feature properly works in production, an integration test is executed that needs to be coordinated between multiple teams that each deliver their piece:

> *[C1a] "A lot of communication is required to ensure that each piece of the puzzle is available at the same time in the test environment to ensure that we can properly test the new feature",* Integrator

The interdependencies require extensive coordination between teams, which is performed by an integrator role. The integrator role however experiences a lack of influencing power while coordinating the Scrum team activities. The lack of influencing power makes coordination much more labor intensive and difficult:

> *[C1b] "Scrum teams are stimulated to close the hatches and concentrate on and realize what is on top of their backlog….. I'm drowning in coordination activities*

because teams need to do it themselves while they are only focusing on their own team", Integrator

Prioritization: One of the key difficulties is having the activities properly prioritized on the product backlog in each of the codependent Scrum teams. Only if the priorities of all codependent teams match, a front to back feature can be delivered. However, priorities are often mismatched. Since an integrator is responsible for delivering a front to back feature he gets stuck in case of mismatching priorities as he has not much influencing power to change the priority in one of the backlogs:

[P1a] "The product owner does not want to cooperate in delivering inter-dependent functionality…. I do not have any influencing authority", Integrator

The responsibility of the integrator however brings expectations that have to be fulfilled. The integrator therefore tries to fix the front to back prioritization which is considered an exhausting exercise:

[P1b] "Each of the Scrum teams needs to have the required activities on their backlog, such as having the test environment available and determine test cases…. To achieve this cost an tremendous amount of energy", Integrator

Product owners determine their backlog priority based on the set strategic objectives in their management line. However, the management lines have different strategic priorities leading to mismatched backlog priorities:

[P1c] "The managing directors in our board have different priority settings. The managing director would like to improve his business process, while the manager of the Internet Channel needs to resolve a number of compliance issues and the manager of Marketing and Sales want to develop software for new product offering. They all think that their objective will be achieved", IT development manager

Such mismatches leads to a situation that each Scrum team develops software for its own product owner, while at the end of the sprint nothing works front to back.

A shared design between teams helps to determine the backlog items that enable a smooth delivery of features. Architects and groups of designers therefore play a significant indirect role in the prioritization process:

[P1d] "The design is very important, created by the architects. But also lead developers. For instance architects define the services and interfaces between the applications of the Scrum teams", IT integrator

Alignment: One of the alignment issues is the difference in deadline between Scrum teams. In case teams work towards a shared deadline less coordination is required. However in case the deadlines differ, a team that is ready needs to repeat activities in the next sprint, while the product backlog of that team has a different focus by that time:

> *[M1a] "If one team is ready and the other team still needs to develop their functionality, the team that is ready needs to store their functionality and has to process new changes before feature testing starts. In that case the discussion starts all over again for the team that was ready, because the flow differs between the teams",* IT manager

Alignment issues are in particular perceived during front to back feature testing. Testing is a complex, labor intensive and highly interdependent exercise. During testing teams need to jointly prepare and execute front to back feature testing, requiring the alignment of the interdependent activities, definitions and terminology.

> *[M1b] "A lot of work needs to be redone to synchronize test data to ensure that the correct test data is available in the test environment that needs to be used in the internet channel",* Integrator

Automation: An automation issue concerns the tooling used for status and progress tracking of the backlog items over the chain. The used tool is particularly useful for tracking individual Scrum teams, but not considered useful to track progress of front to back features:

> *[A1a] "Jira is not really useful on chain level. The tool needs to be adapted to make it work on a chain level. It is more an item driven tool",* IT manager

The issue of status and progress tracking is also mentioned by an IT integrator. Next to the suitability of the tool itself the interviewee also mentions the accuracy of the entered information as issue, which hinders coordination activities:

> *[A1b] "Entering information has limited attention. The teams are mainly working on development, administration gets less attention. It is (therefore) very hard to grasp the development progress… They tried to achieve this with Jira. In theory this works fine but in practice it does not work. I have far too limited possibilities to track such progress",* IT integrator

Predictability: One of the key predictability issues is the development of a single software package, by multiple Scrum teams in parallel. The software package was not

designed to support such parallel development, resulting in potential interference between Scrum teams:

> *[U1a] "We have one monolithic software package that is changed by multiple Scrum teams. How do I know that the development in one Scrum team does not conflict with the development that is performed by another Scrum teams. This has to be well thought of",* IT manager

Visibility: The lack of information visibility about the status and progress is also perceived as an issue. Interviewees mention that there is a lack of supporting IT tools that can provide such visibility:

> *[V1a] "Tooling about status and progress is far too limited and too little available at the moment. Such lack makes the work of the integrator very difficult",* Integrator

"Such tools help to keep overview over the development cycle", IT manager

4.4.2 Case study 2: Telecommunication front to back application chain

The telecom company has a centralized IT organization having 34 Scrum teams. The IT organization has a cluster of front-end Scrum teams which develop applications that interact with Scrum teams of Operation that interact with Scrum teams for Billing and Finance. The involved applications are interdependent, implying that a feature can only be delivered by the constellation of front to back applications. We interviewed Product Owners, Scrum Masters, IT development managers and project managers.

Coordination: Also in the telecom case coordination issues are mentioned. Both the project manager and the IT manager recognize the inward team focus instead of a focus on the front to back chain. Such inward focus makes it difficult for boundary spanning roles such as a project manager to motivate collaboration. As a result the team delivers their specific components and not an operational feature at the end of the sprint.

> *[C2a] "I find the biggest challenge still the end-to-end chain. I perceive that a lot of teams still have a component story mentality. But, we have that XML delivered! Great! But in that case you are mentally not working in a chain",*
> IT manager

> *[C2b] "For me it is still difficult to get all those people together. For me it is very important that we have refinement sessions with the front to back product owner and the story owners of each involved team. For instance the story*

owners should provide a coarse estimation of the effort…. It is very important that everybody is available. It has no value to do this with each team separately", Project manager

Prioritization: Priority mismatches between interdependent teams are mostly mentioned in this case (35%). Such priority mismatches are caused by multiple business projects concurrently being executed, while having executives being responsible for their own profit and loss. For instance the front-office, mid-office and back-office each have their own executives, each with their own unique conflicting targets and also the different business lines have their own specified targets

[P2a] "We are executing 15 programs concurrently…. Each program has its executive at business side which has its own profit & loss. So, who determine priority over the chain when each executive wants to have their features with highest priority", IT manager

[P2b] "The minimal shippable product is a real challenge and creates a lot of tension. Because commerce, operation, IT and network all have a different definition what should be completed. Commerce want a many features as possible and operations want to have a stable solution", IT manager

[P2c] "Commerce consumers want their new product on the market asap, while that new product has no value for commerce business and want their new product on the market.", IT manager

For high priority features the needed stories in the interdependent teams are placed as highest priority items on the team backlogs. However each involved executive influences the priority setting of a subset of product owners in the chain, resulting in priority mismatches over the front to back chain. Such conflicts result in impeded delivery of the front to back features:

[P2d] "In team A an end to end feature has highest priority, at team B however a dependent story has low priority due to interdependent stories that need to be delivered first", IT development manager

[P2e] "Sometimes in one team a story has high priority and in the other team the story has low priority while both stories are needed for a feature. In that case a team uses best-effort stories; they think they can deliver this. With a feature this is difficult since at the end nothing is delivered", Project manager

Alignment: We also found alignment issues between teams in this case study, such as difficulties to align the definition of done. A definition of done is defined per team and not aligned over all involved teams:

> *[M2a] "Yes we have a definition of done, but try to align that over the full chain with clear requirements and acceptance criteria. This is still very hard.",* Product owner

Another alignment issue mentioned are the test stubs that functionally differ from the simulated interface. Such differences result in impediments at the end of the sprint that results in failing delivery of a feature.

> *[M2b] "If team A and team B agree that team A creates an interface that sends 'ABC' and team B understands 'ABD' then team A is happy with their delivered 'ABC' functionality. However at the end of the sprint there is a nasty surprise, when everything has been integrated. That is the disadvantage of stubs",* Project manager

Automation: We also found a lack of automation about the tracking of backlog items. This is largely managed by using Excel sheets. Jira is also used but only limitedly on an intra-team level:

> *[A2a] "We mostly use Excel sheets. We have Jira. However there is no proper support on Jira, so this is not properly configured. We want to do this but we already trying to achieve that for 9 months. Even burn-ups and burn-downs are not yet properly configured",* Product owner

> *[A2b] "Using Jira over the chain is very difficult. I'm managing the front-office, with all my teams. Siebel is located in the middle. We cloned the higher level items in Jira to reuse them. However in this case way we disconnected the linkages between teams. If later a story owner said that a new interface was implemented the other team reacted only with… Oh??",* IT manager

Predictability: A small percentage of quotes (5%) concerned unpredictability. The main issue we found concerns the dependencies between teams. In case one team cannot deliver, the deployed feature is delayed:

> *[U2a] "Misalignment in timing between teams happens regularly. Recently I had a feature which was on the list of each involved team. However one of the teams that had delays earlier now experienced test defects, while having a due deadline of the code freeze. At the end of the sprint nothing was delivered", project manager*

Visibility: We found visibility issues that are related to automation. One typical issue is the status of backlog items that is often not visible throughout the chain, because most of the agreements are made informally and not documented:

> *[V2a] "A lot of decisions are made in an informal way, by verbal haggling at the coffee machine. Not much is documented", Product Owner*

4.4.3 Case study 3: Insurance front to back application chain

The insurance company has a centralized IT organization that delivers web application services to the various business lines that deliver the services to the insurance customer. The web applications and user interface components are developed by a cluster of five codependent front-end Scrum teams. When the produced story meets the Definition of Done the product is handed over to a separate integration team, which tests and deploys the developed product to production. We interviewed Scrum Masters and an Epic Product Owner in one cluster.

The cluster of front-end Scrum teams works with a single prioritized backlog. Each of the Scrum teams picks the highest priority story from the backlog and develops the story. Less prioritization conflicts within the cluster of Scrum teams occur, compared to the other two cases, as a result of having a single backlog (see Table 18).

Coordination: Chain coordination is achieved by project managers. One typical coordination issue is the lack of control that project managers have over the Scrum teams. Project managers are used to have full control over project resources. However in the Agile way of working project managers have no direct control, and agree the work with product owners that prioritize the work.

> *[C3a] "For the project manager it is challenging as he needs to get work done from the Scrum teams, while he has no direct influence on the work in the teams, in contrary to the classical waterfall approach in which a project manager has direct control over project resources",* Scrum Master

Coordination issues between Scrum teams are also mentioned. A Scrum team does not have influencing capabilities on the activities of another Scrum team, which results in rigidly limiting accountability to their own team.

> *[C3b] "Classically our stakeholders try to hold us responsible for the front to back chain, including connectivity. However it is unpleasant to be responsible for something that you cannot influence. We therefore make explicitly clear that this is our responsibility and if you want to bring something life other items need to be realized as well",* Scrum Master

Prioritization: Also in this case we found evidence of priority mismatches between Scrum teams caused by conflicting objectives at the strategic level. One Scrum Master states that senior management is simply not familiar with negotiating and decision making:

> *[P3a] "On senior management level managers are unfamiliar with such choices. They are not used to come together with six people and listening to each other's needs and jointly decide on priority",* Scrum Master

In this case a single prioritized backlog over multiple teams is used. However conflicting objectives at the strategic level obstruct an unambiguous prioritization of the backlog:

> *[P3b] "And let us know what we need to start with first. But… sometimes that is not clear. Sometimes there are two conflicting objectives that each needs to be completed first…. At senior management level they are unfamiliar with these kind of choices and sitting at the table together making decisions which goal is accomplished first. They all want that of course",* Scrum Master

Alignment: A typical alignment issue that was mentioned in this case was the different way of working of each of the teams that need to work together to jointly deliver features. To achieve collaboration in such situation is considered difficult:

> *[M3a] "The requirements have been distributed over the different teams, which each needs to adapt their software… the chains however are heterogeneous, with different technologies and a different way of working. During chain integration testing we discover whether everything is working. That is a very tiring process",* IT manager

Automation: Also in this case we found the need for reliable status and progress tracking over the chain of Scrum teams. The company has an integrated suite to automate status tracking, including event handling of the integration software:

> *[A3a] "You need to take care of the work items. To achieve that is a case of discipline, because an engineer programs his lines of code. If he checks in the source, his change is integrated in the code base and the work item is updated. It is all very data warehouse like",* IT manager

Predictability: We found that delivery unpredictability can be caused by misjudgments in the planning. Such misjudgment leads to a lot of extra work at the end of the sprint that can easily lead to an unrecoverable delivery impediment:

> *[U3a] "We did not do something well in the planning. We missed a very important component, which led to straightening out the back-office integration at the last moment",* Scrum Master

To mitigate delivery unpredictability the front-end team uses sequential back to front development, implying that the front-office waits until the required back-office application has been delivered:

> *[U3b] "The process is asynchronous; first there must be a back-office application to be able to define the amount of work",* Scrum Master

Even though the predictability of feature delivery might be increased, such mitigation slows down time to market significantly. For instance a feature that needs to be developed in a chain with four teams requires four sprints to deliver the feature.

Visibility: The found visibility issues concern a factual overview of the backlog items' status in the various Scrum team. Instead of using relative indicators the company is working on making the delivery pipeline transparent to stakeholders, by offering factual reports about the tracking of (requirement) backlog items:

> *[V3a] "His requirement is still in the backlog. He can call us of course, asking when we will start picking up his requirement…. Our management want to understand when something start to happen",* Scrum Master

> *[V3b] "Whatever a red or green smiley is, it is just an undetermined judgment of somebody. And we then start discussing the meaning of such indicators, not about the real progress. So we are now working on making the production line of IT transparent…. The test is 67% succeeded and 33% failed. Two bugs are still open and the progress of a user story",* Scrum Master

4.5 Towards a conceptual model integrating the issues

We identified six issues in chains of codependent Scrum teams: (1) a lack of coordination in the chain (2) mismatches in backlog priority between teams, (3) alignment issues between teams, (4) a lack of IT chain process automation, (5) unpredictability of delivery to commitment and (6) a lack of information visibility in the chain. See also Table 17. The number of quotes counted in the interviews is taken as the weight of the issues.

In the following subsections we discuss each of these issues. We synthesize the empirical results and existing theory to build propositions, which we then use to build the conceptual model. While doing so, we refer back to the related work discussed in

section 4.2. The subsections are sorted in accordance with the design of the conceptual model derived in subsection 4.5.7. The items between brackets '[ ]' refer back to the quotes in section 4.4.

4.5.1 Predictability

The first issue is the unpredictability of whether a feature will be delivered by the interdependent chain of Scrum teams (56x grounded). Each team has to deliver its application functionality for the feature. In case one of the teams does not deliver its functionality, the feature will not be delivered or is at least delayed [U1a, U2a, U3a]. The risk of one team not delivering its functionality increases with the number of codependent teams. For instance if each team delivers the necessary functionality in 9 out of 10 cases and the chain consists of 10 codependent teams, the chance of a feature delivered in the sprint is $(0.9)^{10}$, which is less than 35%. In case a feature is not delivered in a sprint, all involved teams need to spend time on that feature in the next sprint. The team that did not deliver needs to manufacture the functionality in the next sprint and the teams that did deliver need to do rework, such as retesting and bug fixing. Unpredictability is therefore considered a major cause of cost increase and longer time to market. Yet, in the existing literature as discussed in section 4.2 predictability is mentioned implicitly or the Scrum framework is asserted as having a positive impact on predictability (Dove & LaBarge, 2014).

4.5.2 Coordination

We identify lack of coordination between Scrum teams as the most often mentioned issue in chains of Scrum teams (124x grounded). The lack of coordination expresses itself during the full development lifecycle [C1a, C2b, C3a]. In the first case study an integrator role coordinates the activities between Scrum teams [C1b] and in the second and third case business project managers take care of these coordination activities [C2b, C3a]. In all three cases these coordinating activities are performed by an actor (integrator or project manager) which is not part of any Scrum team. Each of these interviewed roles perceives a lack of influencing capabilities. The interviewees mention that the Scrum teams only focus on their own backlog and not on delivery of (front to back) features [C1b, C2a, C2b, C3a, C3b, P1a]. The lack of influencing capability results in unpredictability whether a feature gets delivered.

Scrum of Scrum meetings are typically used to coordinate activities between Scrum teams and issues with these Scrum of Scrum meetings do occur, as discussed in subsection 4.2.2 (Paasivaara et al., 2012). However, hardly any issues with regard to these meetings are mentioned by the interviewees.

Coordination manages the interdependent activities between the codependent Scrum teams (Malone & Crowston, 1990; Sharp & Robinson, 2010). Coordination theory helps

to understand the deeper concept of coordination. Coordination theory is described by Malone and Crowston (1990) and defined as: *"a body of principles about how activities can be coordinated that is, about how actors can work together harmoniously"*. The components of their definition of coordination are: (1) goal identification, (2) activity mapping on goals, (3) actor assignment on activities and (4) interdependency management. The Scrum framework contains these components on a Scrum team level. For a chain of Scrum teams however these components are not supported by the Scrum framework, while they are of critical importance.

Scheerer, Hildenbrand, and Kude (2014) use coordination theory to compile a conceptual framework in Agile settings with three forms of coordination: (1) mechanistic coordination - coordination by plan or rules with little communication, (2) organic coordination - coordination by means of mutual adjustment or feedback via interaction, which can be formal and planned or informal and spontaneous and (3) cognitive coordination - based on explicit and tacit knowledge the actors have about each other, such as a shared mental model (Mathieu et al., 2000). Their conceptual framework shows that coordination should be embedded at the organic and cognitive level, next to mechanistic coordination.

Mapping the conceptual framework of Scheerer et al. (2014) with the coordination practices of the three empirical case studies reveals that only mechanistic coordination has been implemented. Organic and cognitive coordination between Scrum teams is hardly existent due to the explicit Scrum policies, such as the focus on the team specific backlog, fixed sprint cycles and predefined roles, with limited focus on front to back coordination. The mechanistic form of communication seems also ineffective, since coordination is performed by a coordinator role that has limited mandate. We argue that coordination practices should be deeply embedded within and over the Scrum teams by implementing the components from Malone and Crowston (1990) on a Scrum chain level, while utilizing all three forms of coordination as proposed by Scheerer et al. (2014). Deeply embedded coordination will result in better coordination between teams resulting in increased predictability in each of the teams and therefore more delivery predictability. We therefore propose:

> ***[P1]*** *Embedded coordination practices within and between Scrum teams positively impact delivery predictability*

4.5.3 Prioritization

Priority issues have been identified by multiple researchers (Begel et al., 2009; Lehto & Rautiainen, 2009; Petersen & Wohlin, 2009; Waardenburg & van Vliet, 2012), as discussed in section 4.2. However the related work is unclear about the research environment and interpretation of the research results is therefore difficult.

In our case studies, mismatched backlog priority is considered the second largest issue in a Scrum chain (122x grounded). Priority of a backlog is set by a product owner. The product owner sets his priority based on the input the product owner receives, typically derived from the strategic objectives. The goals however differ at strategic level [P1c, P2a, P2b, P2c, P3a]. For instance the goals of the front-office function (e.g. sales) naturally differ from those of a finance function. When each product owner sets his backlog priority based on their (strategic) goals [P1a, P2b, P2c], prioritization over the front to back chain gets mismatched. Mismatched priority setting makes each Scrum teams concurrently developing different functionality for different features [P1b, P2d, P2e], and at the end of the sprint likely no feature gets delivered. Priority mismatches consequently result in increased delivery unpredictability, leading to our next proposition:

> ***[P2]*** *Matching priority over the front to back chain positively impacts delivery predictability*

Prioritization is a way to set goals (Gregory et al., 2011; Locke & Latham, 1990), while goal setting is one of the components of coordination theory (Malone & Crowston, 1990). Matching priority over the front to back chain implies a single goal for all Scrum teams, embedding one of the coordination theory components. We therefore propose:

> ***[P3]*** *Matching priority improves front to back coordination practices*

We found additional related literature in the area of strategic decision making. Strategic decision making is a bounded rational process (Eisenhardt & Zbaracki, 1992). The bounded rational process is caused by the cognitive limitations and motivational and emotional factors (Bazerman & Moore, 2009). These cognitive limitations result in a biased perception of reality and biased decision making.

A decision maker is also influenced by social and political factors (Boonstra, 2003). For instance the business manager (strategic decision maker 1) might be put under pressure to reach a sales goal and therefore gives a risk mitigation program lower priority, while the chief risk offer (strategic decision maker 2) might have the strategic goal to lower overall financial company risk. The product owner of the front-office Scrum team will likely be put under pressure to prioritize the backlog in accordance with the sales goal, while the product owner of the finance Scrum team will likely prioritize in accordance with the risk reduction goal.

We argue that priority matching over the front to back chain will be improved, in case the priorities match at the strategic level. Such strategic priority matching requires the implementation of strategic decision making strategies, such as (1) using decision-analysis tools, (2) acquiring expertise, (3) debiasing judgments, (4) analogical

reasoning, (5) taking an outsider view and (6) enhancing the understanding biases of peers (Bazerman & Moore, 2009; Eisenhardt & Zbaracki, 1992). Supported by bounded rationality theory and decision making strategies we argue that:

> ***[P4]*** *The implementation of decision making strategies improve matched priority setting*

4.5.4 Alignment

Another issue is the misalignment between the codependent Scrum teams (99x grounded) [M1a, M3a]. Although Scrum has a prescribed structure, the working processes can be implemented in many different ways, leading to such misalignments.
An identified misalignment is the definition of done [M2a]. One team defines 'done' as delivered before system testing and another team defines 'done' as delivered including system testing. A second identified issue is the misalignment of the start, the finish and duration of the sprints. One Scrum team has a two week cycle and another team has a monthly cycle. A third identified issue is the misalignment of test activities and test results between Scrum teams [M1b, M2b].
The need for alignment is also reported by Saddington (2012) (see section 4.2). However Saddington (2012) targets the alignment between product owners which is covered as priority matching in our case study.
The misalignment between codependent Scrum teams causes unpredictability and delivery delays. For instance if the sprint of team B ends two weeks later than team A the delivery of the feature is delayed until team B has completed its sprint. The delivery even gets blocked in case team A is has no feature testing item on the backlog of their next sprint. Because when team B is ready to test their functionality, team A is not available for testing. We therefore propose:

> ***[P5]*** *Alignment between Scrum teams positively impacts delivery predictability*

The Scum framework does not support alignment between Scrum teams. A Scrum team will optimize its working processes for its own purposes. Only in case the teams have an interest in an aligned way of working, and are able to understand and influence each other's way of working, alignment is enabled. Supported by coordination theory we argue that (1) a common shared goal and (2) a coordination mechanism improve alignment between Scrum teams.
(1) The key goal of a Scrum team is to add business value, which is one of the control theory components (Malone & Crowston, 1990). With codependent feature delivery the value is realized by codependent Scrum teams. The primary goal of a Scrum team should therefore be to realize a front to back feature, instead of realizing top priority items on their backlog. The goal on realizing features will face team members with

misalignments and motivate them to proactively align their way of working. We therefore propose:

> ***[P6]*** *Matched priority setting positively impacts the alignment between Scrum teams.*

(2) A proper coordination mechanism should be implemented as conceptualized by Scheerer et al. (2014). Mechanistic coordination is achieved by the implementation of policies. Organic coordination can be achieved by implementing communities of practice (CoP), such as a test CoP with the objective of aligned front to back testing. Cognitive coordination is achieved by implementing a shared mental model and transactive memory (Jonker, van Riemsdijk, & Vermeulen, 2011; Lewis & Herndon, 2011; Wegner, 1987). Misalignment in the shared mental model between teams causes misunderstanding and misinterpretation (see also Hildenbrand et al. (2008) in subsection 4.2.2). The more of the mental model is shared, the more cohesive the chain can operate (Mathieu et al., 2000). Given these three contributing types of coordination we propose:

> ***[P7]*** *Coordination practices positively impact the alignment between Scrum teams*

4.5.5 Visibility

We also identified the lack of information visibility as an issue (38x grounded). Interviewees mention the need for visibility over the prioritized backlogs, the development status of a feature and the underlying functionality [V1a, V2a, V3a]. Such information enables teams to take mitigating actions and manage expectations. A lack of visibility disables teams to take appropriate action, which leads to uncontrollable impediments later in the sprint. An example illustrates the need for visibility: a Scrum team needs to realize a backlog item for a feature. After a week the realization of the item is impeded, putting feature delivery at risk. In case codependent teams have visibility over the backlog they become aware of the impediment and can take the necessary mitigating actions.

Control theory theorizes the role of visibility in the chain and the positive impact on coordination practices between the Scrum teams. Although control theory is historically used as a mathematical model to explain the behavior of physical systems, the basics can be also applied to human actors (Andrei, 2006; Vlietland & van Vliet, 2014b; Wiener, 1965). Control theory consists of three fundamental concepts. The first one is goal setting, which in this case is set by the prioritized backlog items. The members of Scrum teams take action to realize the backlog in case of sufficient level of

visibility over the set goals; see also Saddington (2012) in section 4.2. The second concept of control theory is feedback. Feedback informs the actors about the actual status and progress of codependent teams, again in case of sufficient visibility. The third concept is the comparison function that compares the actual value with the goal. The difference between the two values is fed into the action process, resulting in adapted interdependent action by the team members in the codependent teams.

Looking from a visibility perspective reveals that the components of coordination theory, as described by Malone and Crowston (1990), are strongly related to the concepts of control theory. Both theories have the components goals and (interdependent) activities. Goals are used to direct the activities and the coordination activities are used to achieve the goals. We therefore argue that visibility is an essential ingredient for effective coordination practices. For instance, coordination practices are impeded without proper levels of visibility over the goals, because the activities cannot be directed towards the goal. For the same reason a lack of visibility over the interdependent activities impedes coordination practices. Information visibility therefore enables coordination practices, leading to the proposition:

[P8] *Information visibility positively impacts coordination practices*

4.5.6 Automation

Automation is the sixth identified issue (72x grounded). One of the mentioned automation issues concerns the lack of backlog status and progress information of the codependent teams in the tracking tool [A1a, A1b, A2a, A2b, A3a]. The lack of such automated information sharing hinders the necessary mitigating activities [A1b, A2b] in the Scrum teams. Automated information sharing of item status and progress falls under Agile Continuous Planning (see subsection 4.2.3), although we found no related work about the lack of such information in Agile settings.

We use related literature in the area of Supply Chain Management to build our last proposition. Supply Chain Management (SCM) is the management of interconnected network, channel and node businesses involved in the provision of product and service packages required by the end customers in a supply chain (Harland, 1996). Workflow in supply chains, which is similar to workflow in Scrum chains, is managed by means of coordination rather than centralization (Mentzer et al., 2001).

Coordination in supply chains requires the visibility of operational (e.g. status and progress) information (Datta & Christopher, 2011; Wei & Wang, 2010). Visibility of such information is enabled by information technology, which is widely used in the area of supply chain management. Given the similarity of supply chain management with a chain of Scrum teams we argue that automation will also benefit visibility in our Scrum chains, which leads to our last proposition:

[P9] *Automation of status and progress tracking in the chain positively impacts information visibility.*

4.5.7 Conceptual model

We use the propositions to construct the conceptual model shown in Figure 23. Each proposition connects two issues. For instance the proposition [P9] connects the two issues: (1) '*Automation*' and (2) '*Visibility*'. Each of these two issues is represented as a rectangle in Figure 23. The rectangle, that represents the issue *'Strategic Decision Making'* has a dashed line because the issue was solely grounded in theory (see section 4.5.3) and not identified as an issue through the empirical cases.
An arrow represents the relationship between two issues. The direction of the arrow indicates the dependency between the two issues. For instance the issue '*Automation'* positively impacts the issue '*Visibility'*, as explained in subsection 4.5.6.

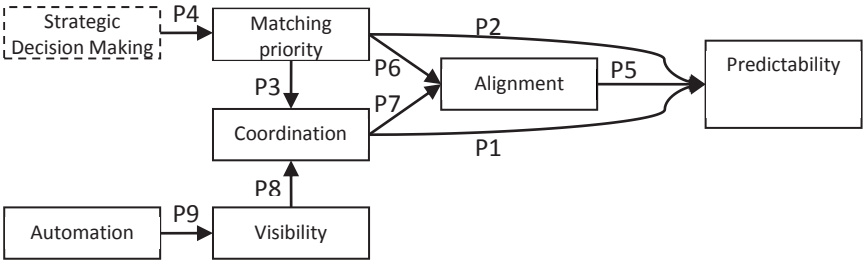

Figure 23, Resulting conceptual model

The conceptual model can serve as starting point for the development of a governance framework to mitigate the identified issues in chains of Scrum teams. We propose a framework that extends the existing Scrum framework with roles that have authority and accountability over the front to back chain. The framework should include a set of coordination and alignment processes for managing front to back feature delivery. But governance is more (A. E. Brown & Grant, 2005):

> *"IT governance is not about what specific decisions are made. That is management. Rather, governance is about 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. Good IT governance draws on corporate governance principles to manage and use IT to achieve corporate performance goals".*

The framework should therefore also include decision making processes to allow matching priority over the front to back chain. For each strategic decision making it should be clear (1) who is the decision making authority over priority setting, (2) who provides input about a decision and (3) how these roles are jointly held accountable (A. E. Brown & Grant, 2005).

The governance framework potentially fulfills the need for structure as mentioned by Talby and Dubinsky (2009), Soundararajan and Arthur (2009) and Batra et al. (2010) (see subsection 4.2.1). Since the governance framework might lead to less Agility in the IT development center, it should comply with the Agile manifesto (Beedle et al., 2013) by having the right mix of plan-based and agile strategies (Batra et al., 2010; Port & Bui, 2009; Soundararajan & Arthur, 2009). The existing models of Kniberg and Ivarsson (2012), Leffingwell (2010) and Scheerer et al. (2014) can be utilized as starting principles for such governance framework.

4.6 Threats to validity

We used case selection criteria and multiple cases to enhance research rigor and external validity. The external validity can be improved further by studying and comparing additional cases in the same area and in other areas, such as IT incident handling chains (Vlietland & van Vliet, 2014c) and service supply chains (Baltacioglu et al., 2007).

For content validity purposes we grounded the interview questions in the dimensions of Galbraith and 3C following the definitions of Sharp and Robinson (2010). We furthermore tested the actual involvement of the standard Scrum Master and Product Owner roles in the Scrum chain and we interviewed supplementary roles where applicable. Bias in the coding and aggregation process was reduced by analyzing the root-causes of the quantitative differences and similarities between the cases (see Table 18). The result of that analysis led to improvements in the coding and aggregation process.

We triangulated the empirical research results by studying archival records, conducting 18 in-depth interviews, conducting verification interviews by phone and using field experts to verify the results.

The inductive nature of the research has several research limitations. First our six issues were identified in three cases, while the preliminary analysis based on two cases identified four issues. Additional case studies might lead to additional issues and extensions or alterations in the conceptual model, since the conceptual model is based on the six identified issues.

Second the qualitative nature of the analysis always results in some levels of analysis bias, even though we used cross-case and quantitative analysis for triangulation. Third the interviews will not collect information that the interviewees consider not necessary to share or consider too sensitive to share.

4.7 Conclusion

We identified six issues in chains of codependent Scrum teams: (1) a lack of coordination in the chain (2) mismatches in backlog priority between teams, (3) alignment issues between teams, (4) a lack of IT chain process automation, (5) unpredictability of delivery to commitment and (6) a lack of information visibility in the chain. The synthesis of these issues with existing theory resulted in nine propositions. These nine propositions have been combined to a conceptual model.

The results show that the application of the Scrum framework in an interdependent application chain can be challenging. Several issues result in unpredictable feature delivery, while the Scrum framework provides little support to mitigate such issues in Scrum chains. Coordination is perceived difficult as coordination is largely allocated to individual coordinators that have little mandate. Priority mismatches result in different Scrum teams developing functionality for different features in parallel. To enable matching priority on operational level the priority needs to be matched at the strategic level at the first place, while priority setting at the strategic level is influenced by emotional, sociological and political factors. The Scrum framework also offers little guidance on the alignment of working processes between Scrum teams. We also found that a lack of automation and visibility of the status and progress of codependent backlogs items impedes collaboration in the chain.

The results imply that the application of Scrum in an interdependent application portfolio needs to be governed. We propose a governance framework to manage chains of Scrum teams in the enterprise that addresses the identified issues, while complying with the Agile manifesto. The framework should include decision making processes for matched priority setting. As a distributed context hinders collaboration between teams, the governance framework should include tailored support for distributed teams, depending on the applied distribution model Sutherland et al. (2007).

Our future research avenue is to empirically test the conceptual model in an existing chain of Scrum teams and to elaborate the conceptual model towards a governance framework that supports the mitigation of the six issues.

A second avenue is to empirically research the strength of the causal relationships between the issues, as well as alternative ways to measure the strength of the issues. A third future research possibility is to analyze additional cases and refine the set of issues and the conceptual model. Another possible future research avenue is to apply the conceptual model in other chains, such as the service supply chain industry.