Raoul Vallon[1], Christopher Dräger[1], Alexander Zapletal[2], Thomas Grechenig[1]

1: Vienna University of Technology, 2: Research Industrial Systems Engineering R&D

Adapting to Changes in a Project's DNA: A Descriptive Case Study on the Effects of Transforming Agile Single-site to Distributed Software Development

Problem Statement

- Agile processes have been designed for collocated teams

- We can see increasing research effort to apply Agile values to distributed software development (DSD) environments as it is a current real-world problem

- Adaptations become necessary

This study contributes to the research field by analyzing a transformation from a single-site to a distributed development environment over the period of 15-months in an Austrian IT organization.

Case Study Design: Objective

- *Descriptive* case study: attempts to explore and explain, does not have the capacity to fully understand cause and effect

- Enhance the body of knowledge of agile DSD with a focus on the transition from a collocated to a distributed Scrum process

RQ: *What are the effects of scaling Scrum from a single-site to a DSD environment and how can challenges be overcome?*

Case Timeline

R&D

- 24 months
- Regular Scrum (single site)

Decision Making

- 3 months
- GO/NO GO for Product Development

Product Development

- 15 months
- Agile Distributed Software Development

Before Case Study → Case Study

Case Background: Project's DNA

	R&D (before case study)	Product Development (case study period)
Development sites	1	2
Team members	9 (1 product owner, 1 scrum master, 6 developers and 1 testers)	19 (1 product owner, 2 scrum masters, 11 developers and 5 testers)
Time	24 months	15 months
Project Goal	Technology proof of concept and field test	Market-ready product (new feature development with focus on quality, performance and operational stability)
Process model	Scrum	Scrum extended for agile DSD

Case Background: Distributed Environment

Distributed Sites	Dev	Test	SM	PO	Sum
Development Site	11	0	1	1	13
Test Site	0	5	1*	0	6
Overall	11	5	2	1	19

*Action Researcher

Goal: form integrated teams over two sites (within same city)

Case Study Design: Action Research

- Action Research (AR) is "a circle of planning, action, and fact-finding about the result of the action" (Lewin, 1946)

- Rigorous action research approach by defining a theoretical framework in advance as suggested by Blichfeldt and Anderson (2006)

- Our declared-in-advance framework consists of a priori propositions, which have been developed using the following approach:
  1) Analyze assumptions for collocated Scrum
  2) Find violations of these assumptions in DSD environments

Case Study Design: Action Research

1) Analyze assumptions for collocated Scrum (Strode et al. 2012):

- Synchronization activity (daily scrum, planning, retrospective)
- Synchronization artifact (board, sprint backlog, impediment list)
- Proximity (daily scrum)
- Availability (collocated teams)
- Substitutability (generalist)
- Boundary Spanning Activity (sprint review)
- Boundary Spanning Artifact (board, product backlog)
- Coordinator Role (scrum master)

2) Find violations of these assumptions in DSD environments

- Especially proximity, availability and synchronization are affected
- But also other components need adaptation

Case Study Design: A priori Propositions

We arrived at the following propositions:

P1. *Since agile development has been designed for collocated teams, it* ***has to be extended to work in DSD environments****.*

P2. *Agile DSD* ***cannot achieve the same level of proximity****,* ***availability and synchronization*** *as collocated development. Hence agile DSD does not strive to compete with collocated processes. It aims to find mechanisms to improve coordination by* ***maintaining agile core values*** *such as self-organization and synchronization,* ***while acknowledging assets and drawbacks of distributing software development****.*

Case Study Design: A priori Propositions

P3. *Due to the nature of distributed development, agile DSD needs to* ***rely more heavily on processes, tools and documentation than in its collocated counterparts****. In consequence, balancing the amount of structured and agile elements is a major concern in applying agile concepts to DSD.*

Case Study Design

Unit of Analysis

Occurring problems and decisions taken within 15-months time span

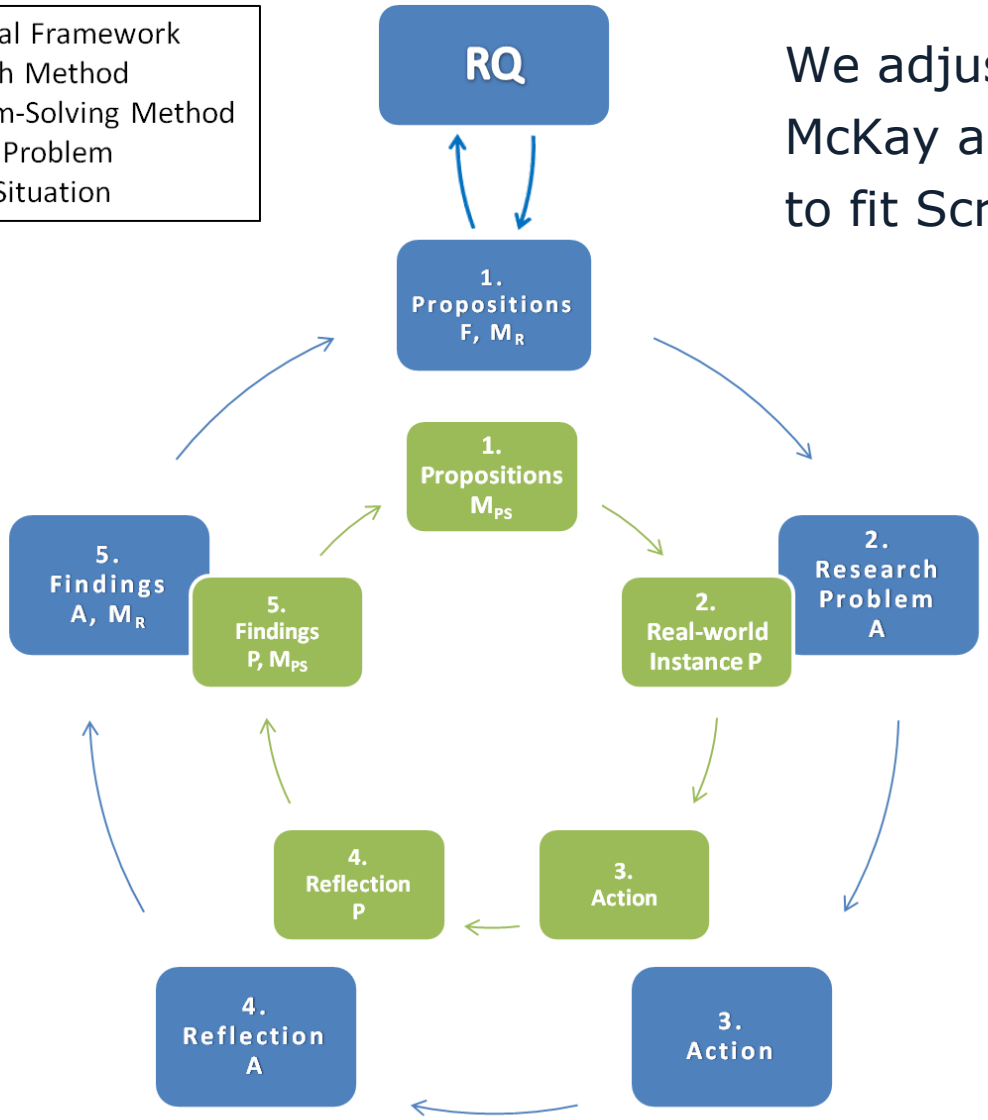
Procedure

Two-cycle AR feedback loop (McKay and Marshall, 2001) with one **problem cycle** to address the problematic situation and one **research cycle** to achieve scientific goals

- Separation of the dual imperatives of AR
- Stakeholders and participants own problem-solving cycle
- Researchers own the research cycle
- Collaboration between participants and researchers to meet respective goals

Dual Imperative AR Cycle: Applied to Scrum

F … Theoretical Framework
M_R … Research Method
M_{PS} … Problem-Solving Method
A … Research Problem
P … Problem Situation



We adjusted the approach by McKay and Marshall (2001) to fit Scrum.

Inner Problem-solving cycle is already part of Scrum.

A second research cycle is built on top of the Scrum process.

Dual Imperative AR Cycle: Applied to Scrum

Cycle Steps	Problem-solving Activity	Research Activity
0. Definition of RQ	-	Define RQ
1. Definition of F,M_R,M_{PS}	(Re-)Define M_{PS} (Sprint)	(Re-)Define F, M_R
2. Update problems	Update problems of P (Daily, Retro)	Update observations of A
3. Take action	Act on problems of P (Sprint)	Collect and analyze data
4. Reflect on success	Reflect on success of actions (Retro)	Draw conclusions for A
5. Update findings	Update findings to P, M_{PS} (Retro)	Update findings to A, M_R
6. Proceed with step 1 until exit criterion is met		

Case Study Design

Data collection

- Project data (beginning of study)
- Project diary (field notes, photos, daily)
- Extract data from issue tracking tool (each sprint)
- Track problems and solutions, record actions (retrospective, each sprint)

Criteria for interpreting findings

The following metrics have been tracked as they were regarded important to the collaboration of dev and test site:

- Bug count
- Release frequency
- Impediments count

Analysis: Kick-Off

27 mostly two-week sprints over 15 months (a few were prolonged to cope with holidays)

Kick-Off: Isolated and Unsynchronized Sites

- Process focused on dev site
- No direct involvement of test site
- Low release frequency
- No definition of done
- Result: test-ready stories instead of deployment-ready ones at the end of a sprint

Analysis: Progress

First Steps towards Cross-functional Teams

- Initial process led to low quality software (sprint 7)
- A full bug fixing iteration (sprint 11) without feature development was necessary to improve quality
- Turning point to follow a more continuous flow

Achievement: Establishing Cross-functional Teams

- Increase contact visits
- Rotate team members towards the end of the sprint
- Mentality for deployment-ready stories achieved
- A more integrated, cross-functional approach was achieved
- Still: non-functional aspects such as performance and operational stability were not valued in teams‘ commitment

Analysis: Progress

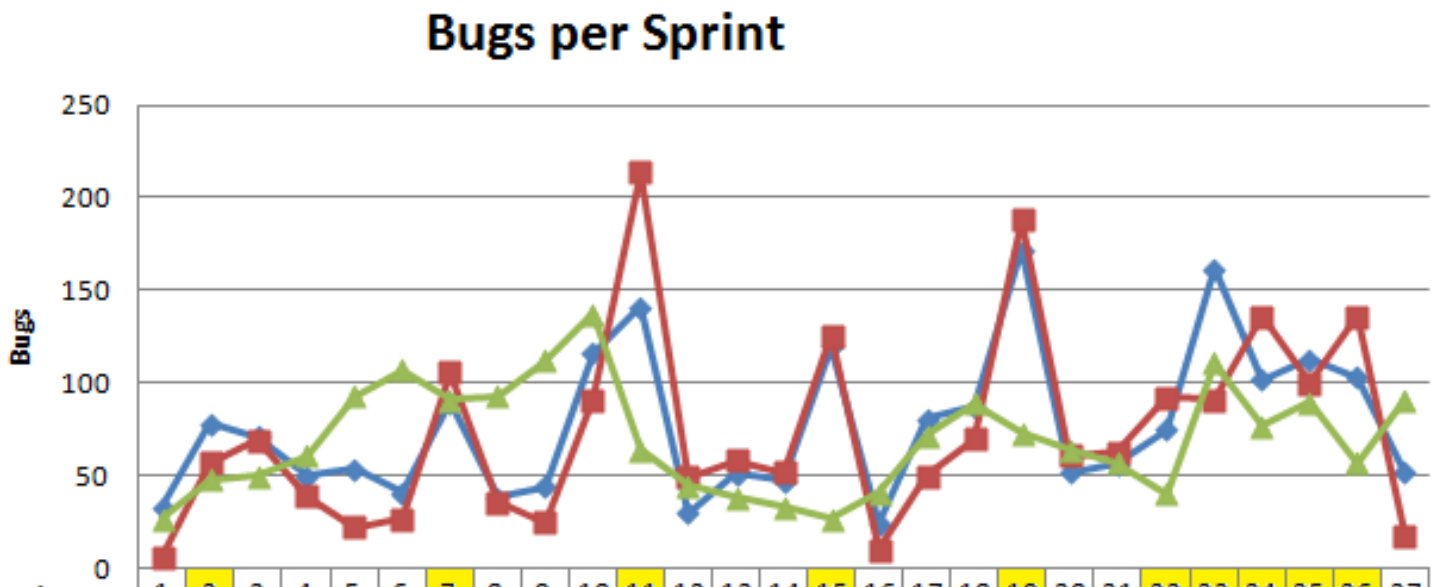
Achievement: Establishing a Continuous Flow

- Bug count and impediment count used to be rather neglected except for sprints with real shipments to the customer
- To force a continuous flow, customer shipments were then scheduled each sprint. This also helped with sprint planning and software quality.

Achievement: Motivation, Willingness to Change

- Retrospective was an invaluable tool for process improvement also during stressful times and also helped to resolve conflicts
- Establishment of trust: Contact visits, online availability, i.e. informal face-to-face coordination was replaced by instant messaging

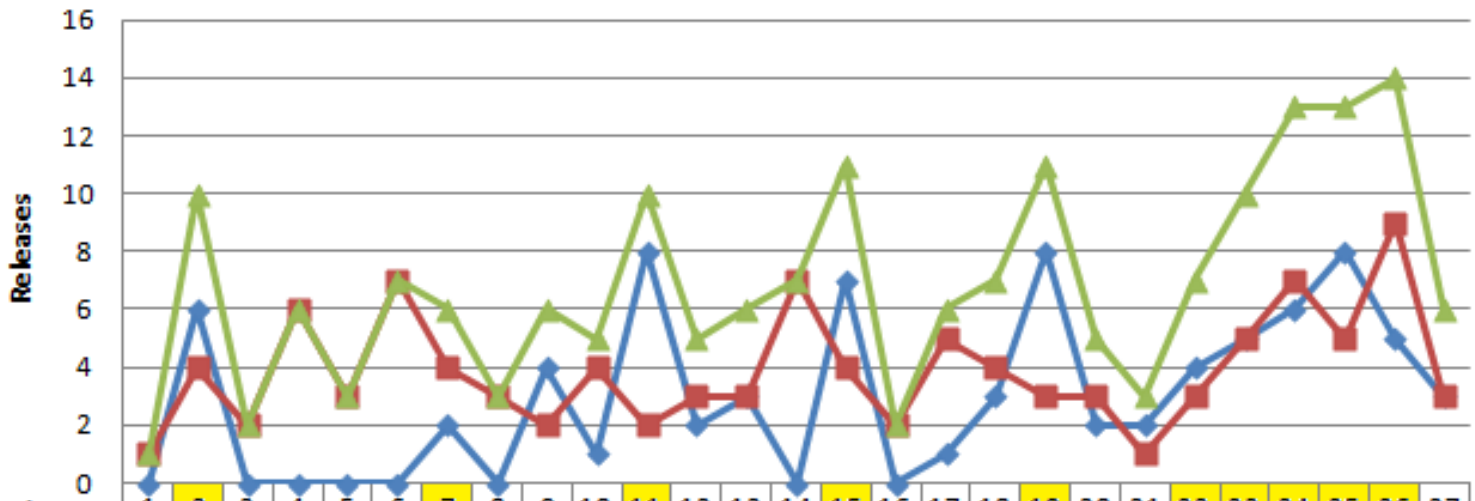
Analysis: Bug Count



Sprints	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
found within sprint	33	78	71	50	54	41	90	38	44	116	141	30	51	47	120	24	81	87	172	52	56	75	162	102	113	103	52
closed within sprint	6	57	69	39	22	27	106	36	25	91	214	49	58	52	126	10	50	70	188	61	63	92	91	136	100	136	18
open at sprint end	27	48	50	61	93	107	91	93	112	137	64	45	38	33	27	41	72	89	73	64	57	40	111	77	90	57	91

Marked sprints denote customer shipments.

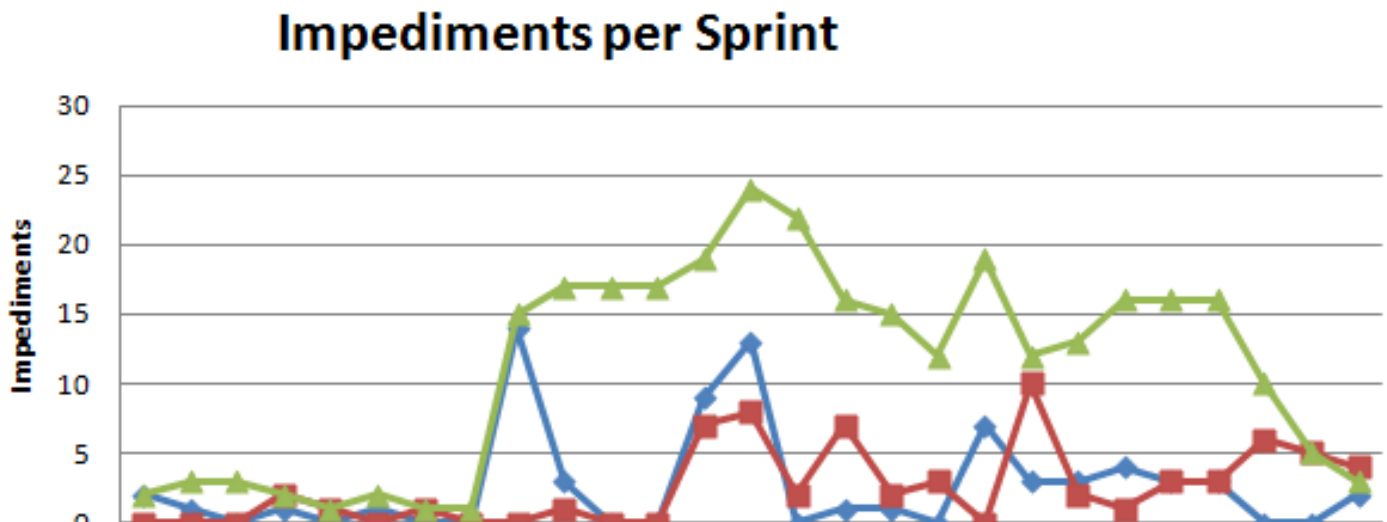
Analysis: Release Frequency



Sprints	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
bugfix release	0	6	0	0	0	0	2	0	4	1	8	2	3	0	7	0	1	3	8	2	2	4	5	6	8	5	3
feature release	1	4	2	6	3	7	4	3	2	4	2	3	3	7	4	2	5	4	3	3	1	3	5	7	5	9	3
both	1	10	2	6	3	7	6	3	6	5	10	5	6	7	11	2	6	7	11	5	3	7	10	13	13	14	6

Marked sprints denote customer shipments.

Analysis: Impediment Count



Sprints	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
new in sprint	2	1	0	1	0	1	0	0	14	3	0	0	9	13	0	1	1	0	7	3	3	4	3	3	0	0	2
resolved within sprint	0	0	0	2	1	0	1	0	0	1	0	0	7	8	2	7	2	3	0	10	2	1	3	3	6	5	4
open at sprint end	2	3	3	2	1	2	1	1	15	17	17	17	19	24	22	16	15	12	19	12	13	16	16	16	10	5	3

Marked sprints denote customer shipments.

Discussion: Agile DSD Learning

RQ: *What are the effects of scaling Scrum from a single-site to a DSD environment and how can challenges be overcome?*

A1. Fully **distributed** cross-functional **teams also need to synchronize and work towards the same goal in a sprint**. Synchronization can be achieved with contact visits, instant messaging, an up-to-date ticket management system and test case or code review.

A2. **Informal face-to-face communication needs substitution in DSD**. **Tool support** can help keep track of changes in requirements. **Instant messaging** can become the equivalent to face-to-face coordination that is available only to collocated teams.

A3. **Continuous customer deployments can help establish focus in DSD** and lead to more realistic sprint plannings and commitments. Continuous flow means more reliable estimations and higher software quality.

Discussion: Agile DSD Learning

*A4. While **up-to-date documentation** does not substitute direct interaction, it **plays a bigger part in agile DSD** than in on-site Scrum since direct communication is harder to manage.*

*A5. **Results** from informal on-site enquiries or meetings **need to be made available to other sites**, either in terms of updated documentation or updated tickets in the electronic ticket management system. Contact visits can further be used to review test cases for critical stories.*

*A6. **Keeping the retrospective as a constant** in an ever-changing process environment provided a valuable asset. The retrospective meeting gathers members from all sites to reflect on the process and discuss improvements. It further functions as an important **mood barometer across all sites**.*

*A7. Process **adaptation in DSD is slower and more difficult to achieve** than in regular collocated Scrum.*

Discussion: A Priori Propositions

P1. *Since agile development has been designed for collocated teams, it* ***has to be extended to work in DSD environments****.*

Evidence in all of A1-A7. Possible extensions include e.g.:

- Instant messaging substitutes face-to-face communication
- Contact visits build and maintain trust
- Use of electronic tools

Discussion: A Priori Propositions

P2. *Agile DSD* ***cannot achieve the same level of proximity, availability and synchronization*** *as collocated development. Hence agile DSD does not strive to compete with collocated processes. It aims to find mechanisms to improve coordination by* ***maintaining agile core values*** *such as self-organization and synchronization,* ***while acknowledging assets and drawbacks of distributing software development****.*

Agile DSD cannot compete with collocated Scrum:

- Slower in adapting to problems (A7)
- Harder to keep team members synchronized (A1)

Discussion: A Priori Propositions

P3. *Due to the nature of distributed development, agile DSD needs to* ***rely more heavily on processes, tools and documentation than in its collocated counterparts****. In consequence, balancing the amount of structured and agile elements is a major concern in applying agile concepts to DSD.*

We experienced a greater need for up-to-date documentation (A4) and electronic tools (A2). It is also important to track results from informal enquiries and meetings (A5).

Case Study Validity and Limitations

Validity

- Declared-in-advance theoretical framework
- Adapted two-cycle AR approach and a construct table
- Reliability: Project diary, saved records of all data, key team members reviewed paper draft

Limitations

- Single case study
- Investigator bias: Reduced by following an adapted two-cycle AR approach and the fact that only one action researcher was on site, so that other researchers remained largely unbiased.

References

Blichfeldt, B.S., Andersen, J.R.: Creating a wider audience for action research: Learning from case-study research. Journal of Research Practice 2(1), D2 (2006)

Lewin, K.: Action research and minority problems. Journal of Social Issues 2(4), 34-46 (1946)

McKay, J., Marshall, P.: The dual imperatives of action research. Information Technology & People 14(1), 46-59 (2001)

Strode, D.E., Huff, S.L., Hope, B., Link, S.: Coordination in co-located agile software development projects, Journal of Systems and Software 85, 6, 1222-1238 (2012)

Thank you