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

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

**Development sites**: The number of development sites went from 1 to 2.

**Team members**: The team size increased from 9 members to 19 members.

**Time**: The project duration was reduced from 24 months to 15 months.

**Project Goal**: The goal changed from technology proof of concept and field test to a market-ready product.

**Process model**: The process model was extended from Scrum to Scrum extended for agile DSD.
**Case Background: Distributed Environment**

<table>
<thead>
<tr>
<th>Distributed Sites</th>
<th>Dev</th>
<th>Test</th>
<th>SM</th>
<th>PO</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Site</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>Test Site</td>
<td>0</td>
<td>5</td>
<td>1*</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Overall</td>
<td>11</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>19</td>
</tr>
</tbody>
</table>

*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
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.*
**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
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

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


Thank you