Aalto University
School of Science
and Technology

Globally Distributed Software Development

Maria Paasivaara

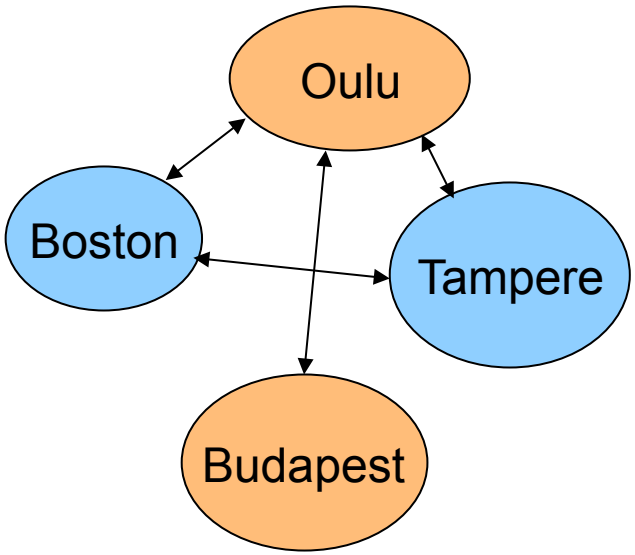
Contents

- Global software development
  - What?
  - Why?
  - Challenges
  - Tips for how to succeed
- Using agile practices in distributed projects

Global Software Development – What?

Global Software Development – What?

- Global software development
- Global software engineering
- Distributed software development (intra/inter-organizational)
- Offshore / nearshore development
  - On-site, off-site
- Software outsourcing
- Software subcontracting



Global Software Development – Why?

Global Software Development – Why?

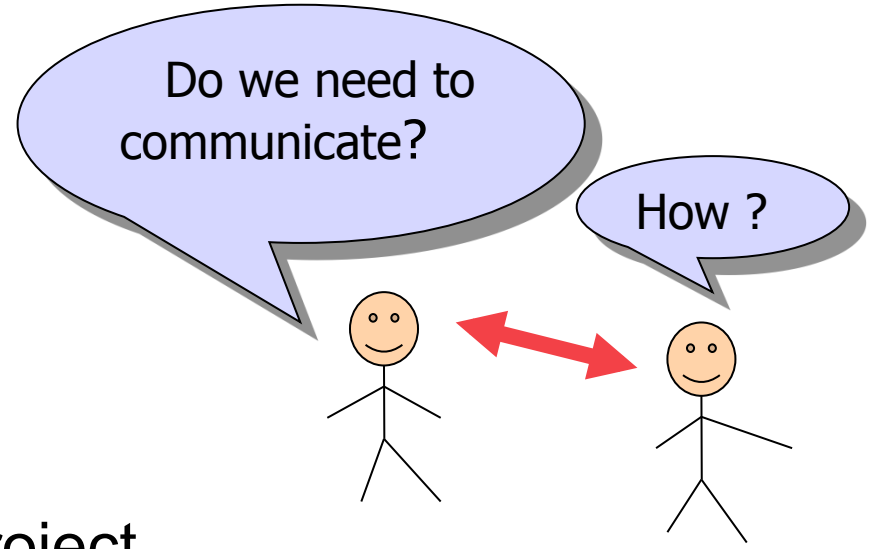
- Concentration on core competences
- Lack of own resources
- Lack of own knowledge
- Increasing the speed of development
- Lower cost (offshore development, nearshore development)
- Development closer to the market
- Flexibility
- Around-the-clock development
- Mergers and acquisitions

What is difficult in distributed software development projects?

Pair discussion

Challenges of Distribution

- **The problem #1: Communication**
- Geographical distance
- Time-zone differences
- Motivational issues
- Cultural differences
- Company border



- Managing a distributed project
  - Takes more time and effort than expected

  --> Distributed projects typically take longer than collocated

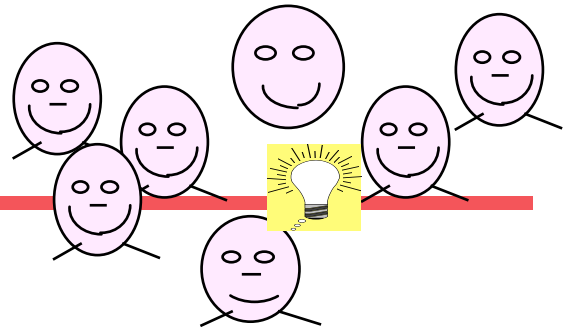
How to make distributed software development projects to succeed?

Pair discussion

Tips for How to Succeed:

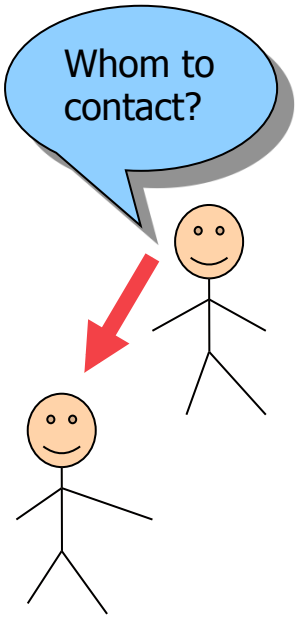
Starting a Distributed Project

- Organization
  - Choose your partners carefully
  - Limit the number of partners and locations
- Minimize the time-zone difference
  - Working across time-zones reduces possibilities for communication
  - Choose sites and partners preferably within same time-zone, if frequent communication is needed



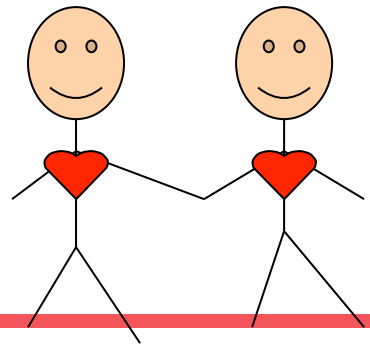
Starting a Distributed Project

- Plan how to divide work effectively
  - Minimize the need for communication between sites – maximize the possibilities for communication
  - Modular product structure
  - Sub-project manager or team leader at every site
  - Maximize the possibilities for communication
- Arrange needed tools: for communication, version/ backlog management, testing, bug repository…
- Plan: What kind of trainings are needed?
  - Hands-on training, class room training….
- Starting a distributed project requires a lot of effort!!
  - Allocate extra time



Team Building and Trust

- Important to build trust between partners and team members already in the beginning
- Plan face-to-face meetings (kick-off, trainings, collocated working periods, developer exchange etc.)
  - Especially meetings in the beginning are important
  - “Give faces” to all sites
  - Seeing good quality work helps to build trust
  - Frequent meetings to ensure good collaboration



Example: Team Building and Trust

In this project testers were reluctant to test the code delivered by a subcontractor from a distant country. The project manager commented:

> *"We had difficulties to get our acceptance testing people to understand that we are in the same boat [with our subcontractors] and it is no use to be enemies. (…) [The reasons behind] might be that when these developers get a delivery and it is not functioning perfectly well, and they know that it is not made by their friends here, but by someone living in Turkey and trying to do it as cheap as possible. (…) And that was the reason why it [testing] was delayed here, because it was not motivating. (…) [In this project] we learned a lot (…) about communication and how much it actually helps to see those [subcontractor's] faces. It was difficult to believe it beforehand!"*

Informing, Monitoring, Transparency

- *"There is a lot of information at the corridors"*
  - In a distributed project people get only the information you give them
  - Do not expect anyone to know anything

    -> make sure they know
- Transparency
  - Give progress information also to team members at all distributed sites - motivation
  - Give feedback to subcontractors / developers at distributed sites - also positive!
- Agree on regular meetings (daily/weekly e.g. teleconference, videoconference)

Example: Informing

In a distributed project there were modules, needed between the parts the customer was developing, and the parts the subcontractor was developing, but nobody was taking care of them. The subcontractor's project manager was quite nervous when he found this out:

> *"We expected that they [the missing modules] would come from somewhere else, until we found the truth. (…) You never know whether some matter is forgotten or whether it is just that they [customer] are not telling us about it. You just don't want many years to ask about things, when you get only counter questions, such as 'How are YOU meeting the deadlines?' – meaning that it is not our business. And then we get feedback that we should carry the responsibility for the whole project. (…) It is hard to be a subcontractor!"*

Example: Give Frequent Feedback

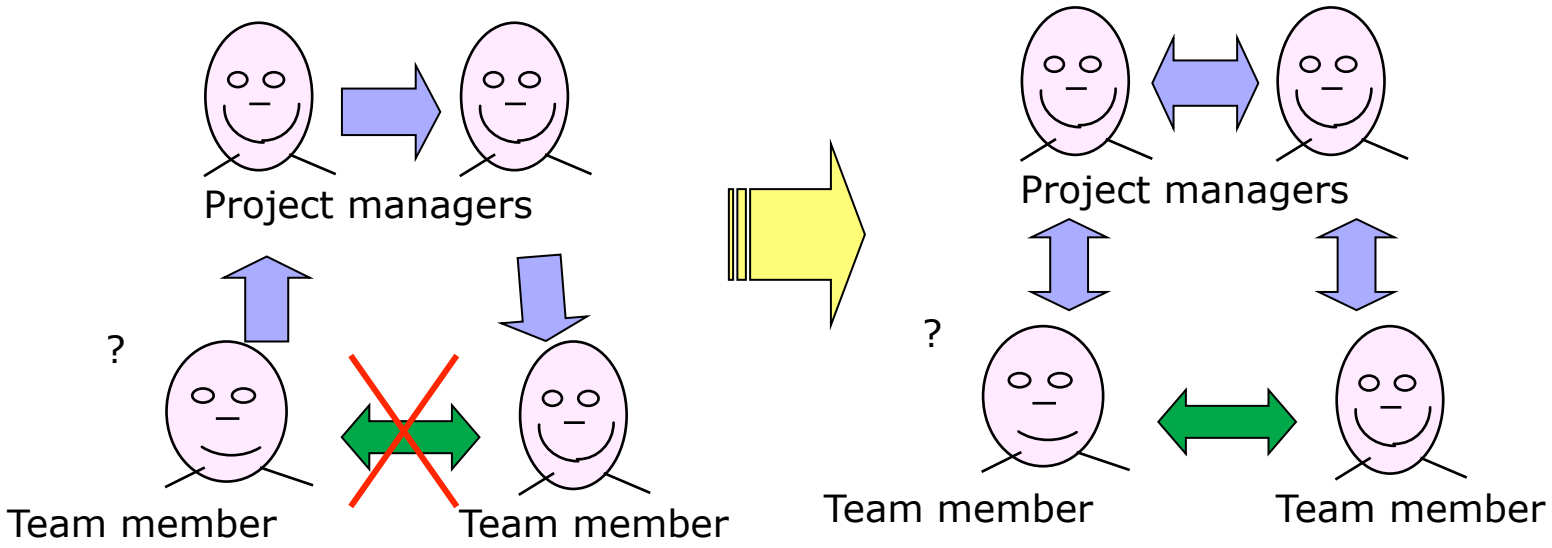
A project manager from a foreign subsidiary commented:

> *"Sometimes I feel that this is like a black hole, we make [code], and send it somewhere. (...) If you haven't got any feedback for your work during the last half year, then you just start to wonder whether it is ok or not. Of course it is not, because when they start to test after that half a year, then you get the mistakes. It is difficult because you have already forgotten what you did a half year ago."*

Continuous Integration, Automated tests

- Arrange continuous integration and automated testing (if not possible daily/weekly deliveries of code & integration)
- Short iterations (e.g. two weeks)
- Benefits
  - Creates transparency
  - Reveals early misunderstandings of requirements
  - Brings real check points
  - Gives instant feedback
  - Adds developer motivation

Project Managers: Bottlenecks or Contact Creators?



Example: Team Level Links

A system architect used a chat program called Messenger to discuss with developers from a foreign subsidiary:

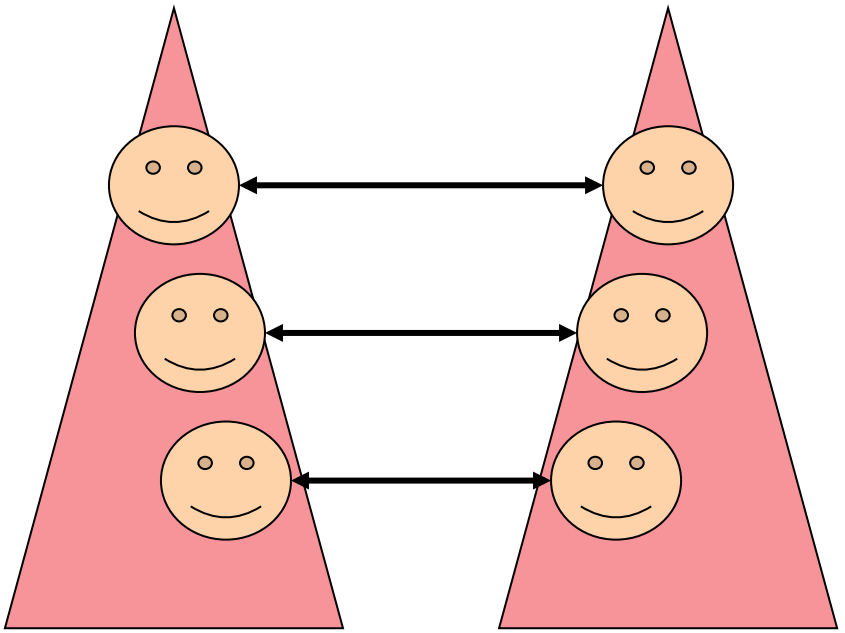
> *"Messenger is more practical than phone, especially with foreign partners. (...) It is already difficult to understand different pronunciations, not to mention the difficulties for me to even express what I want to say. (...) Writing emails takes a lot of time when you have to structure it and give background information. When I write some technical explanation, it takes time, it can take even two hours to write one email when I search information and go back to code. (...) But with messenger, it's more like talking. You do not have to structure things or think too carefully. Comments are very short. You can write about what you have in your mind. And the discussion just flows to the right direction."*

Establishment of Peer-to-Peer Links

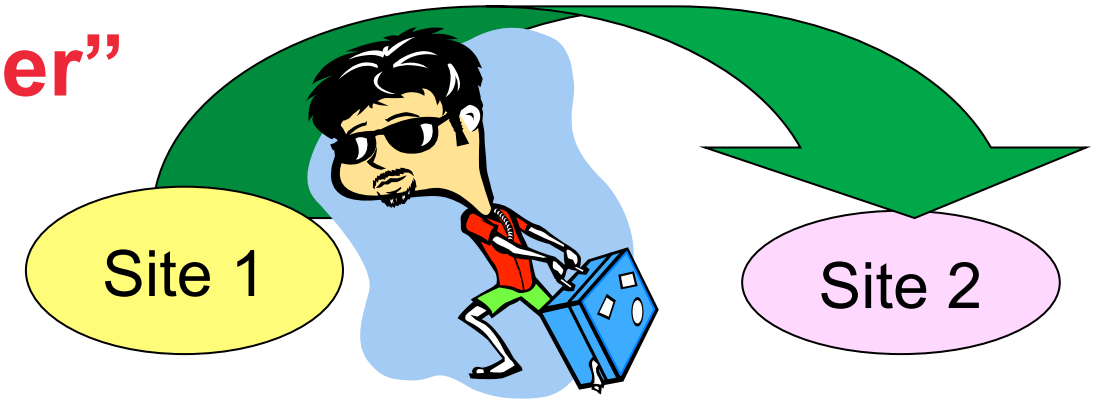
- Establishment of peer-to-peer links between distributed sites / companies at several levels is important

  ⇨ Communication improves

  - Links can be created between 3 levels
    - Management level
    - Project level
    - Team level

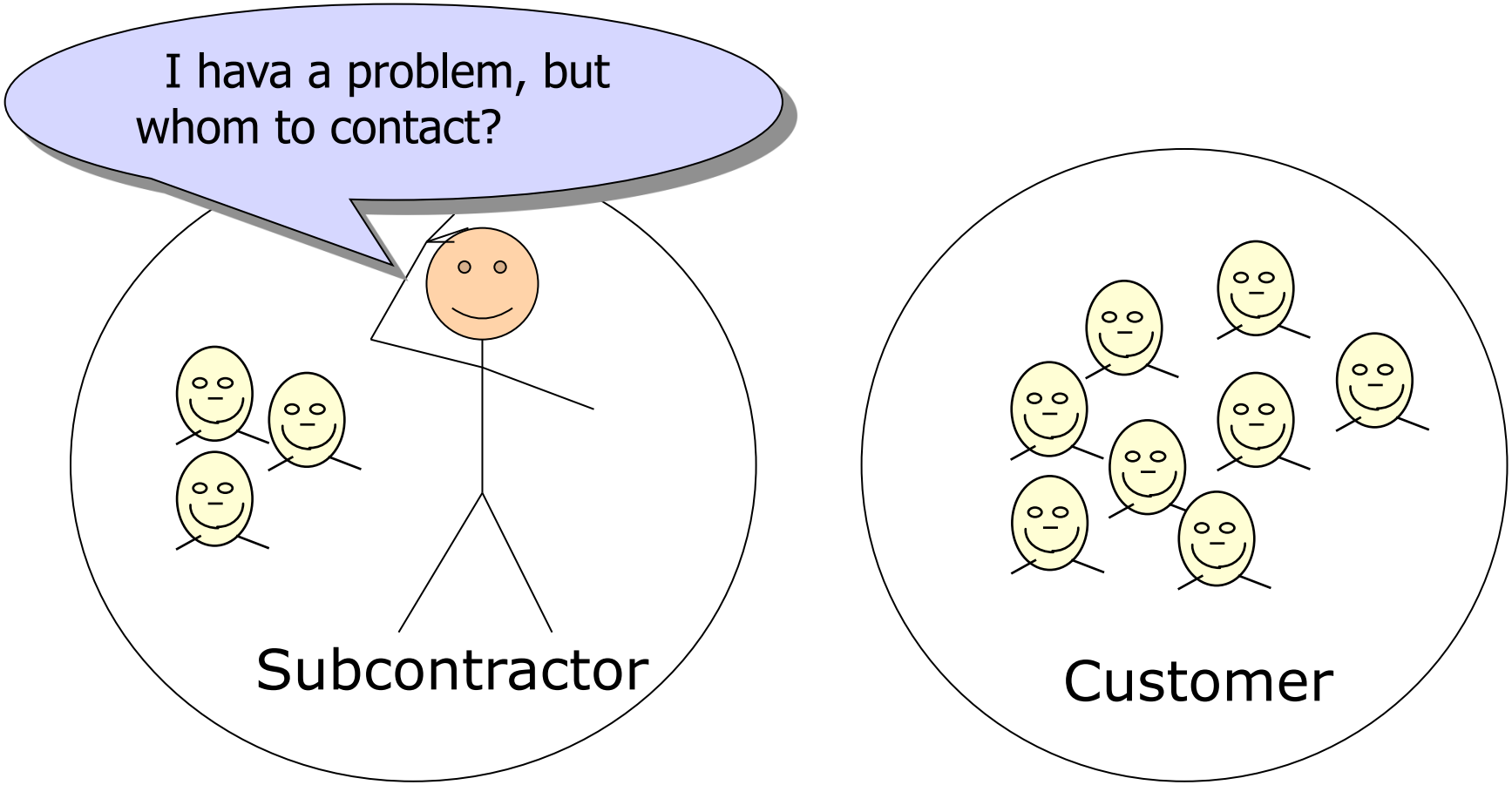


”Visiting Engineer”



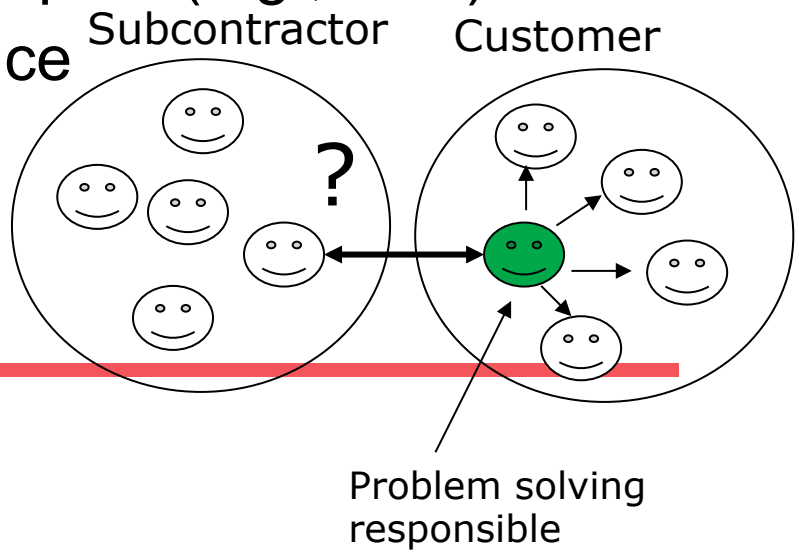
- Visits the collaboration partner (customer, subcontractor, subsidiary, other site)
- Stays and works there for a longer period of time (from one week to several months)
  - Especially in the project initial and final phases
  - In a larger project a small group can visit instead
- One major task is facilitation of communication
  - Passes information, creates contacts, solves problems, is present for face-to-face discussions

Problem Solving



Problem Solving Communication

- Often neglected in the project planning phase!
- Time-consuming
- Important in distributed projects
- The lack of answers delays the project
- Barrier to ask / motivation to answer
- Practices:
  - Direct contacts between developers (e.g., chat)
  - Screen sharing + teleconference
  - Face-to-face problem solving
  - Discussion forums
  - “Problem solving responsible”



Example: Problem Solving

A system architect working in a highly distributed project commented on his new “role”:

> *“It just gradually happened that I became “a link person”. I have a lot of experience since I have been [here] so long. I have also been interested in the big picture of the project. But it is sometimes very hard. Because there are so many tasks, sometimes tens of tasks at the same time. And then I have my own work, what I should develop. (…) My phone rings 40-50 times a day, but at the same time I should code thousands of lines. It is difficult to run from task to task, when you cannot concentrate. (…) Approximately half of my time goes to this kind of communication through phone or email, I’m like a help desk.”*

Using Agile Practices in a Distributed Project

Aalto University
School of Science
and Technology

Combining Agility and Distribution: Why? Why Not?

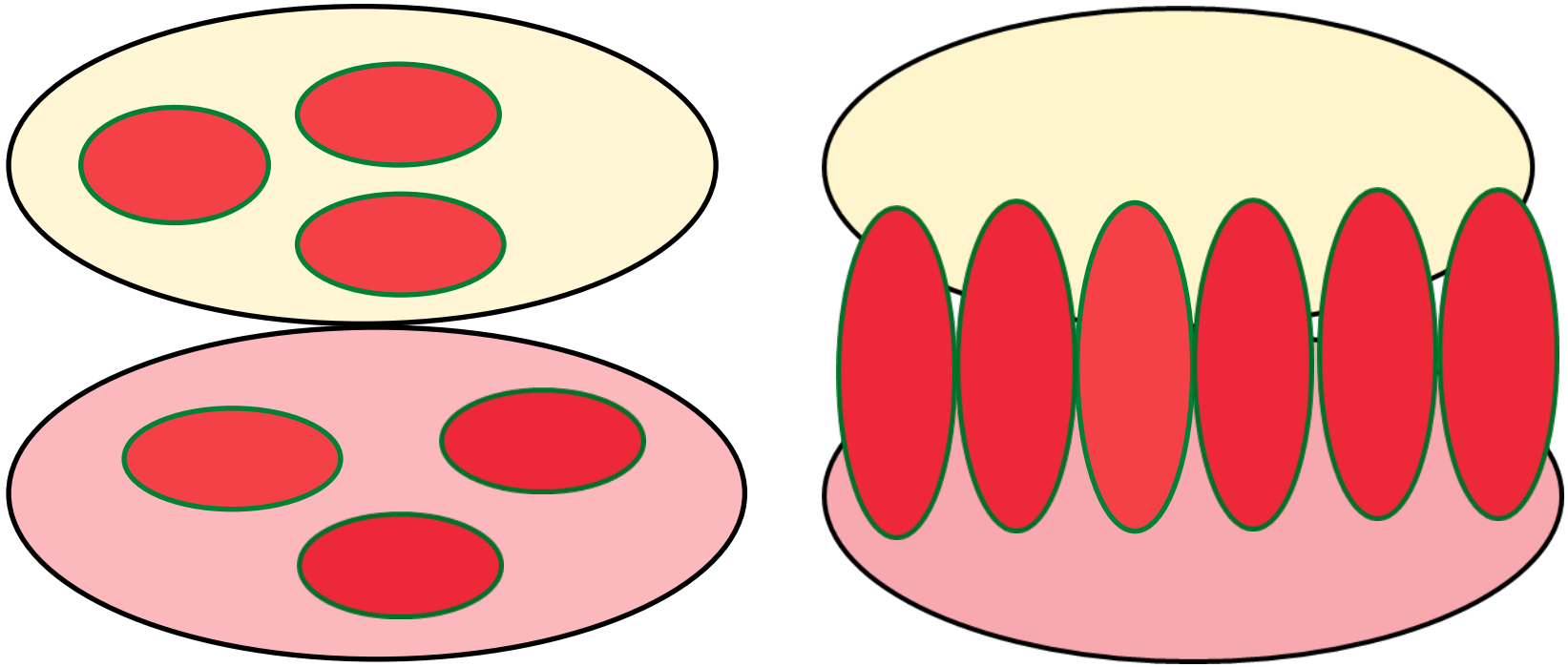
Aalto University
School of Science
and Technology

Distributed Agile Development

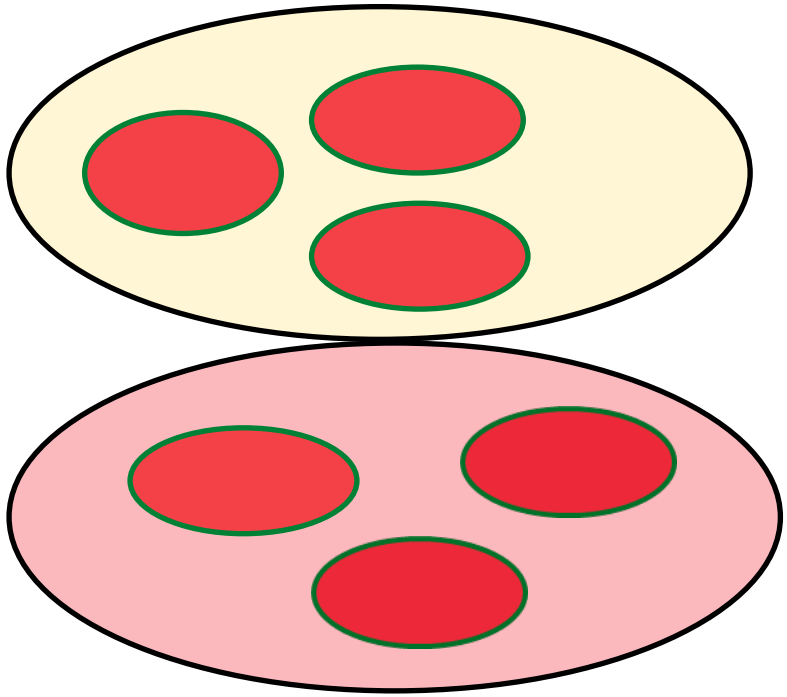
- **Cons:**
  - Agile practices based on collocation and continuous face-to-face communication
  - The biggest problem of distribution: communication
  - Limited experiences in companies

- **Pros:**
  - Agile practices are build on communication: they require everyone to communicate
  - Frequent iterations: fast feedback, response to customer requirements, visibility
  - Many successful experiences: It is possible to succeed!

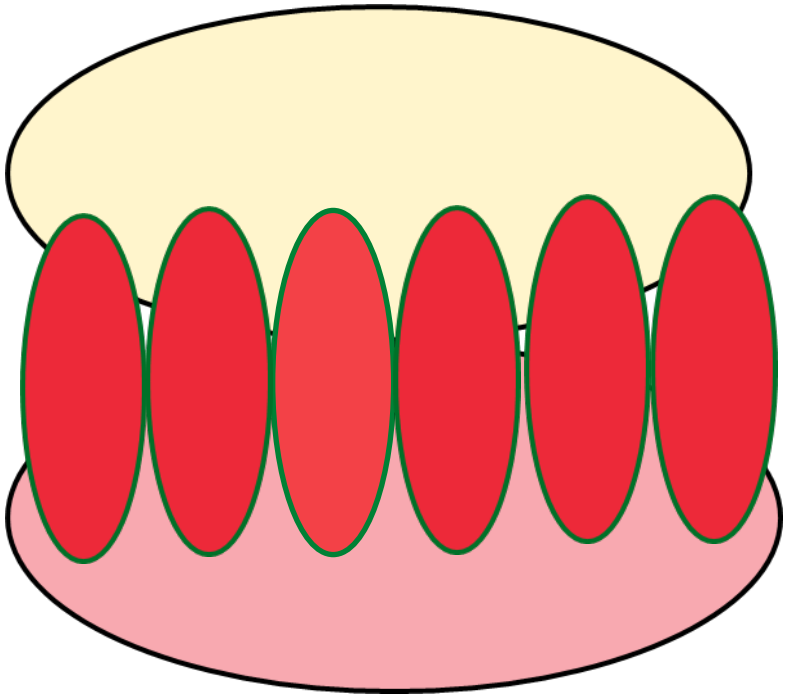
Distributed Scrum Teams?
Site-specific Scrum Teams?



Distributed Scrum Teams?
Site-specific Scrum Teams?



E.g. new product development

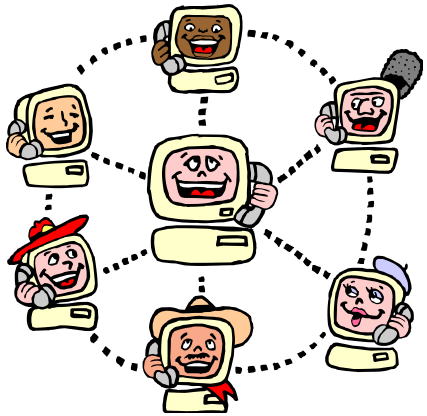


E.g. transforming the development and maintenance of an old product to a new site

Successful Practices for DAD (1/3)

- Daily Scrum meetings of distributed teams (videoconference, teleconference, chat)
  - Requires everyone to communicate – encourage!
  - Peer-to-peer discussions afterwards
  - Scrum-of-scrums

- Continuous communication, e.g. chat
  - Team rooms
  - Videoconference – seeing the faces

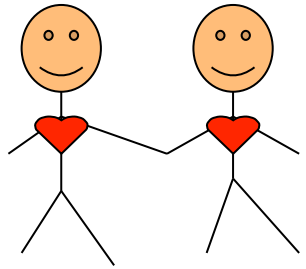


Successful Practices for DAD (2/3)

- Common tools
  - Repository, version management (continuous or daily check-in, nightly builds)
  - Backlog tool (e.g. Wiki, Jira)

- Synchronous iterations (2-4 weeks)
  - Rotating people between teams and sites

- Product owner for each team
  - Participates actively even if distributed!



Successful Practices for DAD (3/3)

- Frequent visits – travel enough!
  - Team building: everyone should meet
  - Hands-on training, collocated working, pair programming etc.
  - Sprint planning together (collocated or distributed)
  - Demo (videoconference, application sharing)
  - Retrospective meeting

Questions?