LUT University

LAND OF THE CURIOUS

LECTURE 12

GLOBAL SOFTWARE ENGINEERING

INFORMATION

- StudentPulse survey
  - Thanks for answering
  - If you did not answer yet, you can still answer until 30.11.!
- Foundations of Information Processing course:
  - Extra lecture for you: tips to exam
  - 7.12. 16-17
  - Zoom & Auditorio 2B

LUT University

COURSE LECTURE SCHEDULE

Teachers:
Maria
Susan
Sami Hyrynsalmi

Date	Topic	Book Chapter(s)
Wed 8.9.	Course introduction	
Tue 14.9.	Introduction to Software Engineering	Chapter 1
Tue 21.9.	Software Processes	Chapter 2
Mon 27.9	Agile Software Engineering	Chapter 3
Tue 5.10.	Requirements Engineering	Chapter 4
Mon 11.10.	Architectural Design	Chapter 6
Wed 20.10.	Modeling and implementation	Chapters 5 & 7
Mon 1.11.	Testing & Quality	Chapters 8 & 24
Mon 8.11.	Software Evolution & Configuration Management	Chapters 9 & 25
Mon 15.11.	Software Project Management	Chapter 22
Mon 22.11.	Software Project Planning	Chapter 23
Mon 29.11.	**Global Software Engineering**	
Wed 8.12.	Software Business	
Mon 13.12.	Project Presentations	

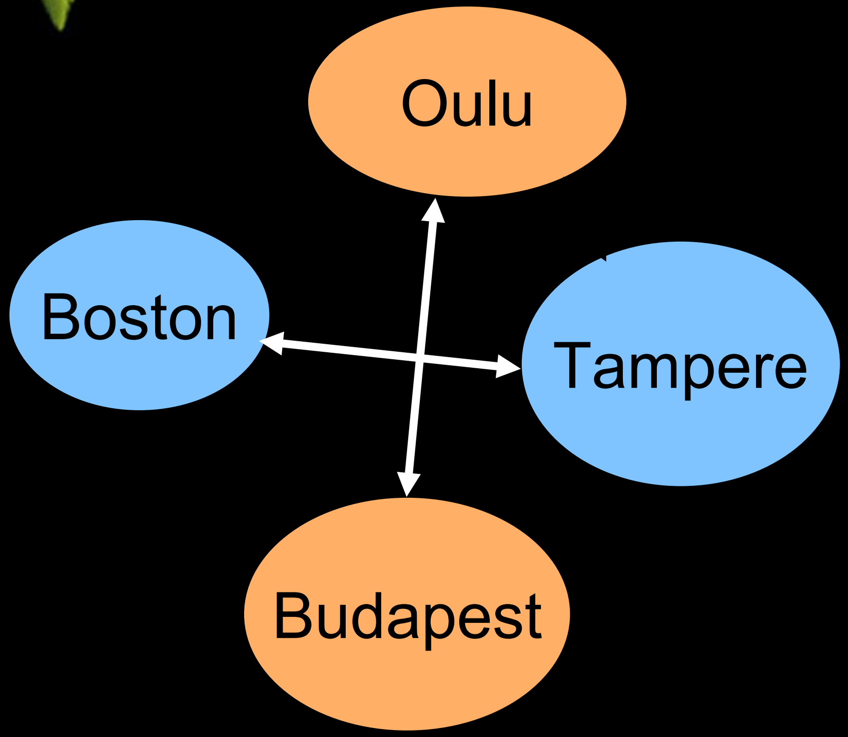
GOALS FOR THIS LECTURE:

After this lecture, you know

1. What is Global Software Engineering
2. Reasons for doing it
3. Challenges of Global Software Engineering
4. Tips to succeed

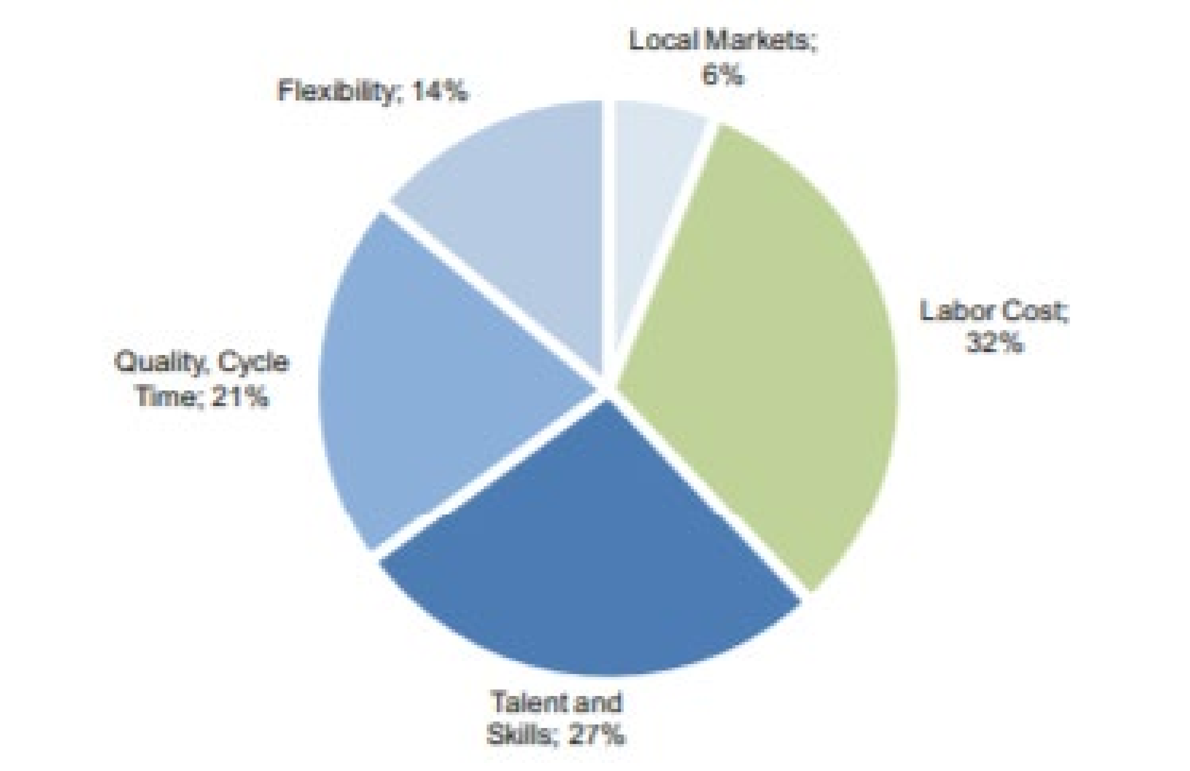
WHAT IS GLOBAL SOFTWARE ENGINEERING?

- *”Software development that is performed in a globally distributed team”*
  - At least two countries involved
- Global software engineering (GSE) can be:
  - Intra- or inter-organizational
  - Off-shore / nearshore development
  - Software outsourcing
  - Software subcontracting



WHY TO USE GLOBAL SOFTWARE ENGINEERING?

- Lower labor cost
- Talent and skills
  - Concentration on core competences
  - Lack of own resources
  - Lack of own knowledge
- Increasing the speed of development
  - Around-the-clock development?
- Flexibility
- Development closer to the market
- Global companies
- Mergers and acquisitions



Source: Ebert et al. (2016) ”Global Software Engineering: Evolution and Trends”, IEEE 11th International Conference on Global Software Engineering

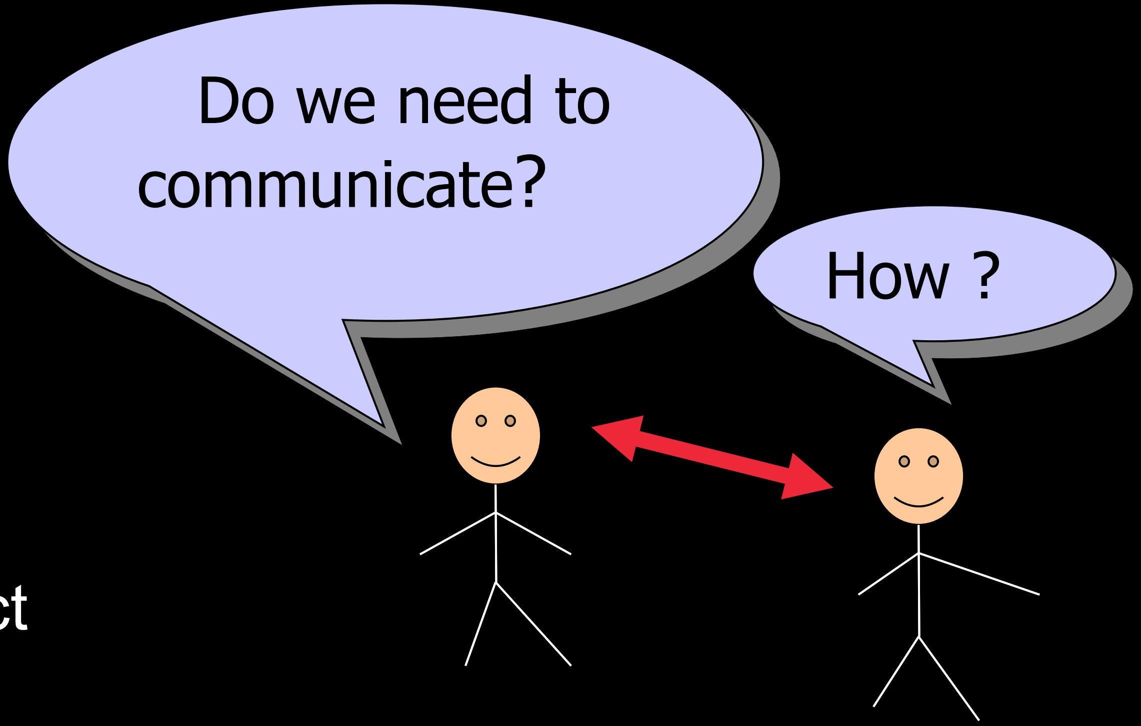
WHAT IS DIFFICULT IN GLOBAL SOFTWARE ENGINEERING?

- Pair discussion

CHALLENGES OF GLOBAL SOFTWARE ENGINEERING

- **The problem #1: Communication and collaboration**
- Geographical distance
- Time-zone differences
- Cultural differences
- Language issues
- Company border
- Motivational issues
- Managing a globally distributed project
  - Takes more time and effort than expected

--> Distributed projects typically take longer than collocated



HOW TO MAKE GLOBALLY DISTRIBUTED SOFTWARE PROJECTS TO SUCCEED?

- Pair discussion

STARTING A GLOBALLY DISTRIBUTED PROJECT

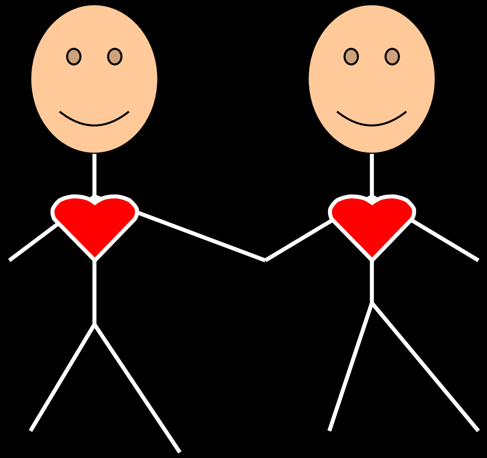
- Organization
  - Choose your partners carefully
  - Limit the number of partners and locations
- Minimize the time-zone difference
  - Working across time-zones reduces possibilities for communication
  - Choose sites and partners preferably within same time-zone, if frequent communication is needed

STARTING A GLOBALLY DISTRIBUTED PROJECT

- Plan how to divide work
  - Minimize the need for communication between the sites – maximize the possibilities for communication – or not?
  - Modular product structure or cross-functional, cross-site, cross-component teams?
  - Sub-project manager or team leader at every site?
  - Distributed or site-specific teams?
- 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 globally 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
  - Frequent meetings to ensure good collaboration
  - Seeing good quality work helps to build trust



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 AND 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 meeting (daily/weekly)

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:

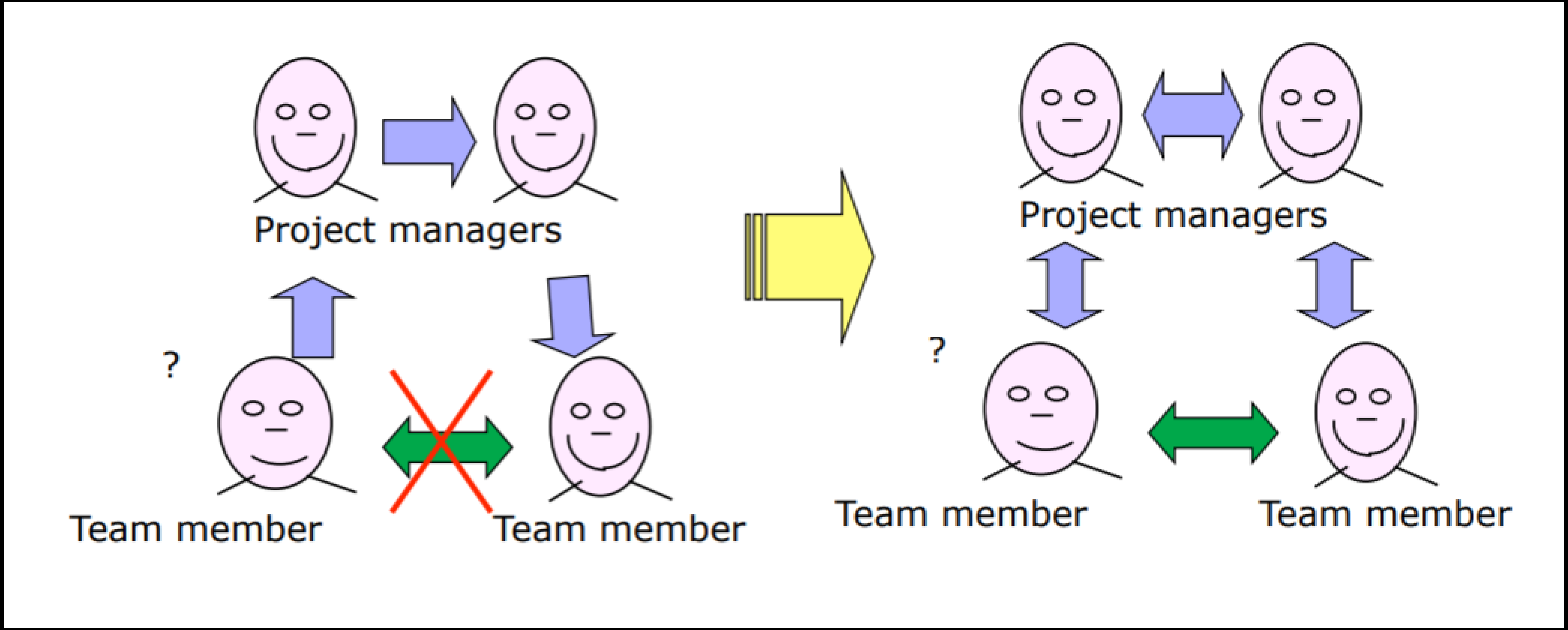
”We expected that they [the missing modules] would come from somewhere else, until we found the truth. (…) You never know whether some matter is forgotten or whether it is just that they [customer] are not telling us about it. You just don't want many years to ask about things, when you get only counter 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."

PROJECT MANAGERS: BOTTLENECKS OR CONTACT CREATORS?

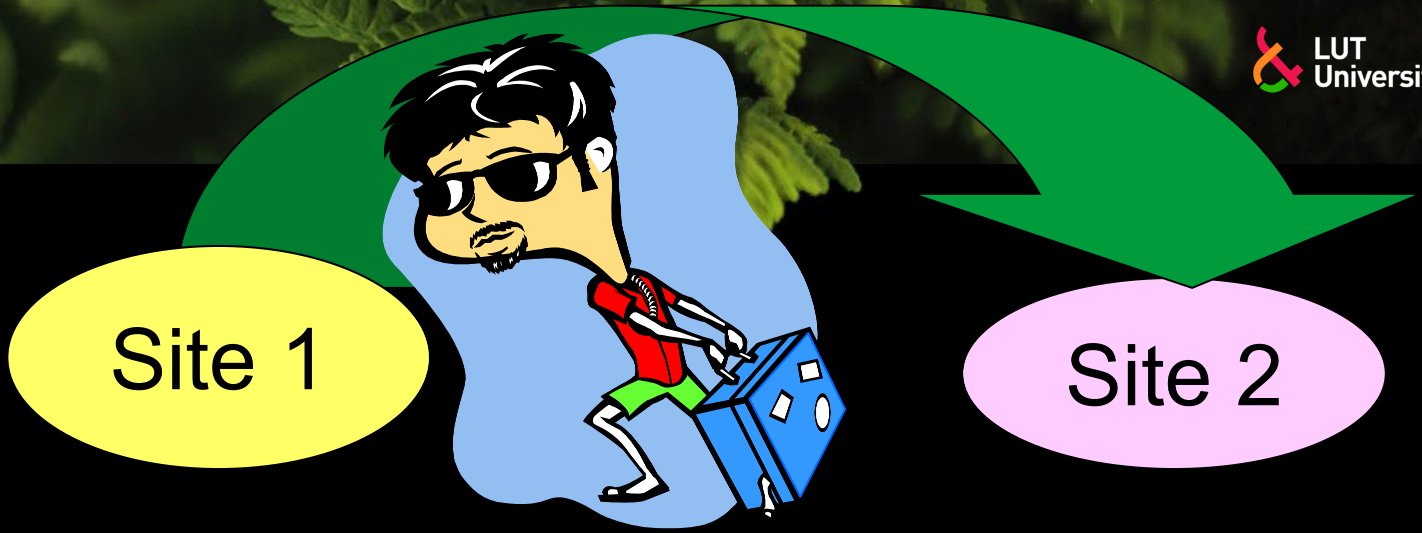


EXAMPLE: TEAM LEVEL LINKS

- 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.”

”VISITING ENGINEER”



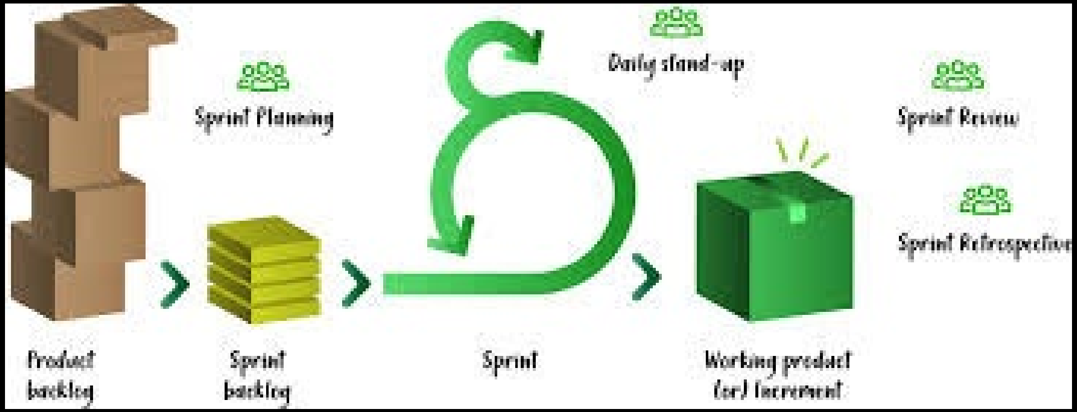
- Visits the collaboration partner (customer, subcontractor, subsidiary, other site)
- Stays and works there for a longer period of time (from one week to several months)
  - Especially in the project initial and final phases
  - In a larger project a small group can visit instead
- One major task is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

AGILE IN GSE – DOES IT FIT?

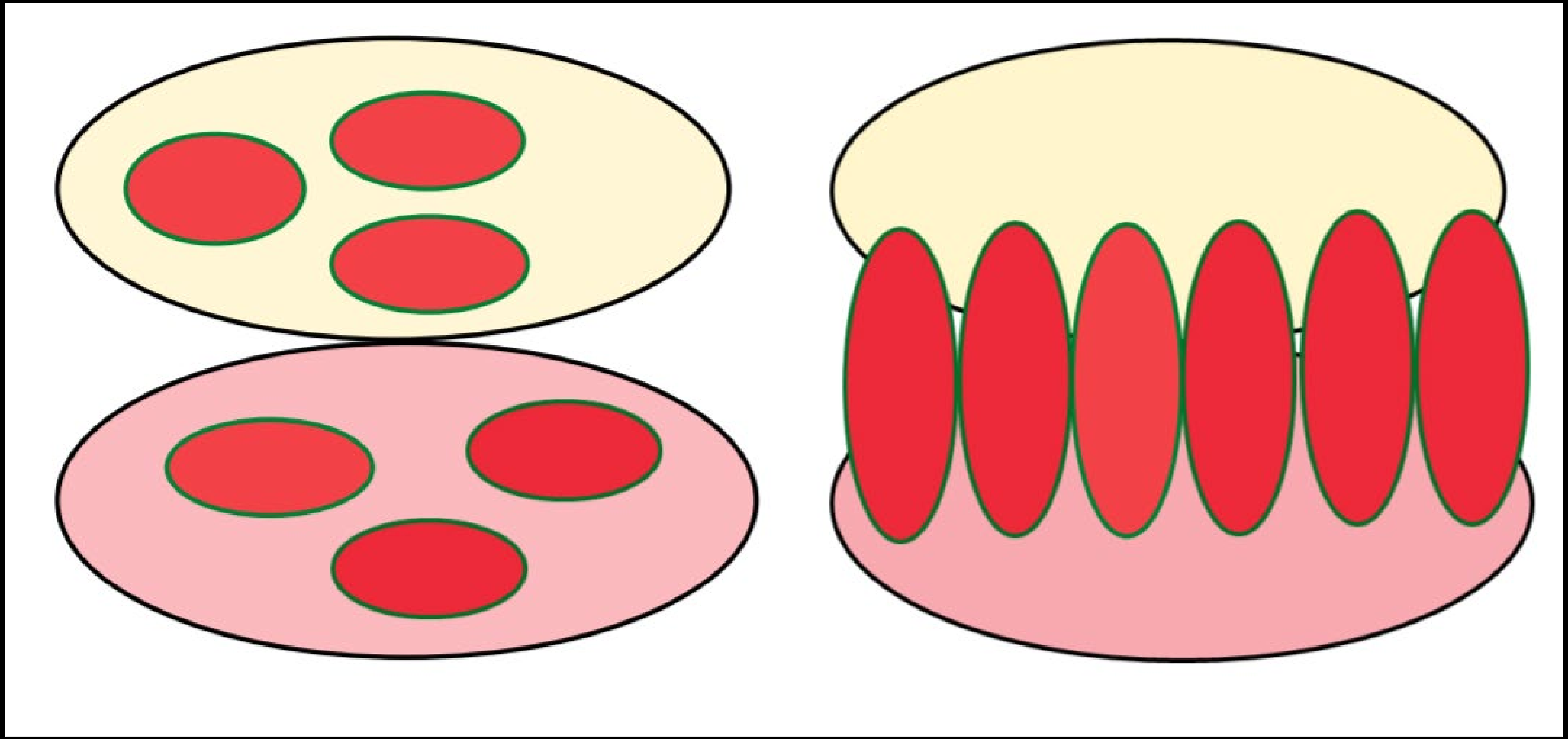
- Cons:
  - Agile practices based on collocation and continuous face-to-face communication
  - The biggest problem of distribution: communication
- Pros:
  - Agile practices are build on communication: they require everyone to communicate
  - Frequent iterations: fast feedback, response to customer requirements, visibility
  - Many successful experiences: It is possible to succeed!

AGILE METHODS, CONTINUOUS INTEGRATION, AUTOMATED TESTS

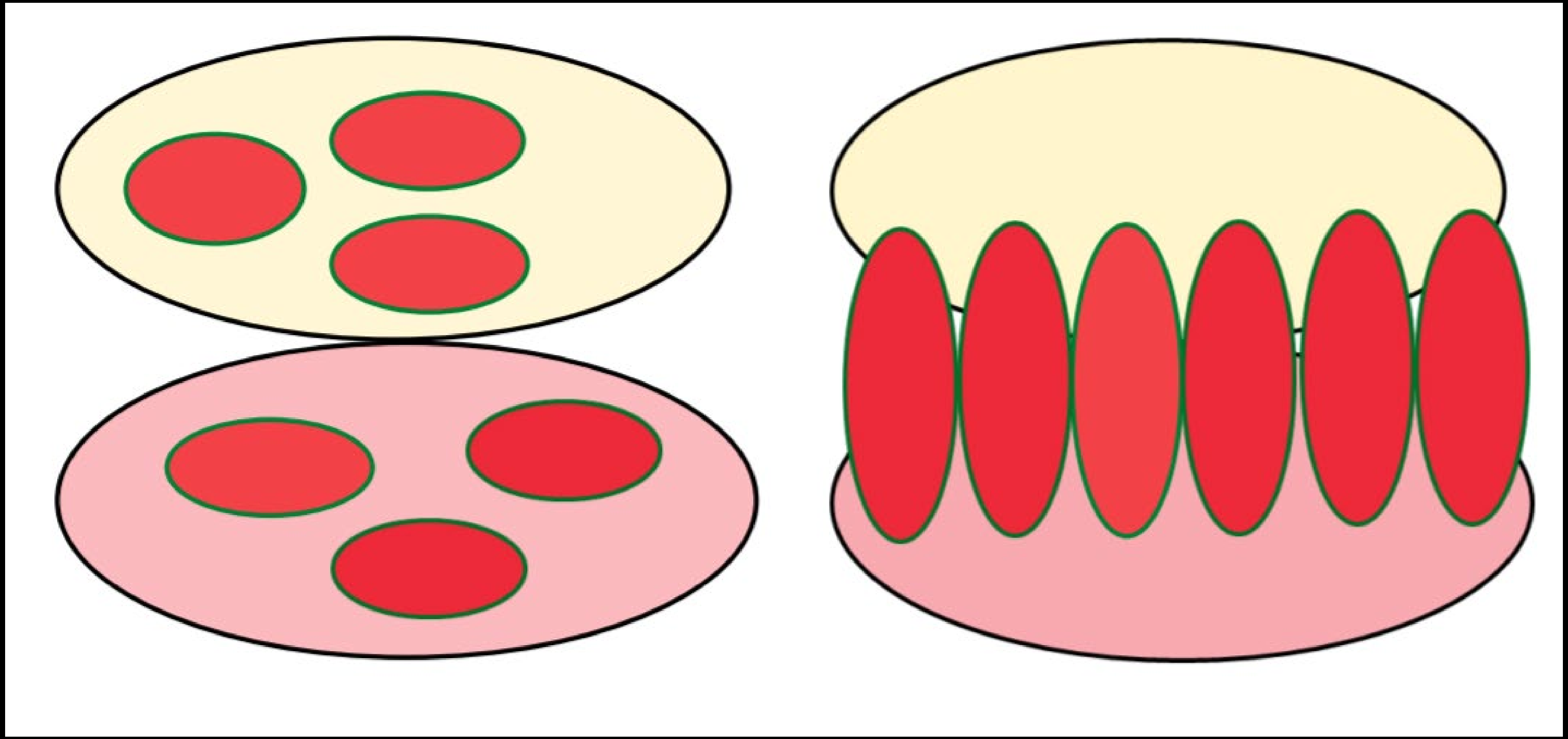
- Agile methods, short iterations (e.g. two weeks)
- Continuous integration and automated testing (if not possible daily/weekly deliveries of code & integration)
- Benefits
  - Creates transparency
  - Reveals early misunderstandings of requirements
  - Brings real check points
  - Gives instant feedback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



DISTRIBUTED OR SITE-SPECIFIC AGILE TEAMS?



DISTRIBUTED OR SITE-SPECIFIC AGILE TEAMS?

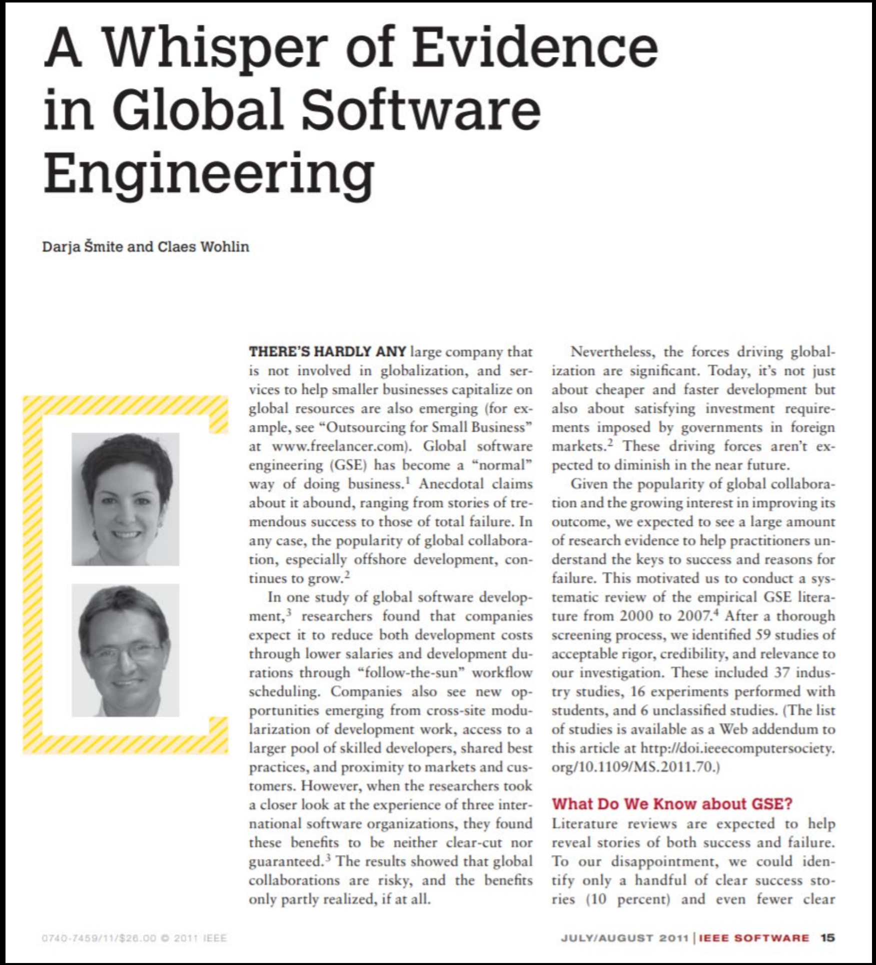


E.g. new product development

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

RECOMMENDATIONS FOR GSE BASED ON A LITERATURE REVIEW

1. Invest in face-to-face meetings, temporal collocation, and exchange visits.
2. Invest in reliable infrastructure, including a centralized repository, common configuration management tools, and rich communication media.
3. Enable effective, frequent communication through synchronous interaction.
4. Keep task dependencies across sites low by implementing decoupled architectural solutions.
5. Implement short incremental development cycles for timely feedback loops.



A Whisper of Evidence in Global Software Engineering

Darja Šmite and Claes Wohlin

THERE'S HARDLY ANY large company that is not involved in globalization, and services to help smaller businesses capitalize on global resources are also emerging (for example, see "Outsourcing for Small Business" at www.freelancer.com). Global software engineering (GSE) has become a "normal" way of doing business.[1] Anecdotal claims about it abound, ranging from stories of tremendous success to those of total failure. In any case, the popularity of global collaboration, especially offshore development, continues to grow.[2]

In one study of global software development,[3] researchers found that companies expect it to reduce both development costs through lower salaries and development durations through "follow-the-sun" workflow scheduling. Companies also see new opportunities emerging from cross-site modularization of development work, access to a larger pool of skilled developers, shared best practices, and proximity to markets and customers. However, when the researchers took a closer look at the experience of three international software organizations, they found these benefits to be neither clear-cut nor guaranteed.[3] The results showed that global collaborations are risky, and the benefits only partly realized, if at all.

Nevertheless, the forces driving globalization are significant. Today, it's not just about cheaper and faster development but also about satisfying investment requirements imposed by governments in foreign markets.[2] These driving forces aren't expected to diminish in the near future.

Given the popularity of global collaboration and the growing interest in improving its outcome, we expected to see a large amount of research evidence to help practitioners understand the keys to success and reasons for failure. This motivated us to conduct a systematic review of the empirical GSE literature from 2000 to 2007.[4] After a thorough screening process, we identified 59 studies of acceptable rigor, credibility, and relevance to our investigation. These included 37 industry studies, 16 experiments performed with students, and 6 unclassified studies. (The list of studies is available as a Web addendum to this article at http://doi.ieeecomputersociety.org/10.1109/MS.2011.70.)

What Do We Know about GSE?

Literature reviews are expected to help reveal stories of both success and failure. To our disappointment, we could identify only a handful of clear success stories (10 percent) and even fewer clear

0740-7459/11/$26.00 © 2011 IEEE

JULY/AUGUST 2011 | IEEE SOFTWARE 15

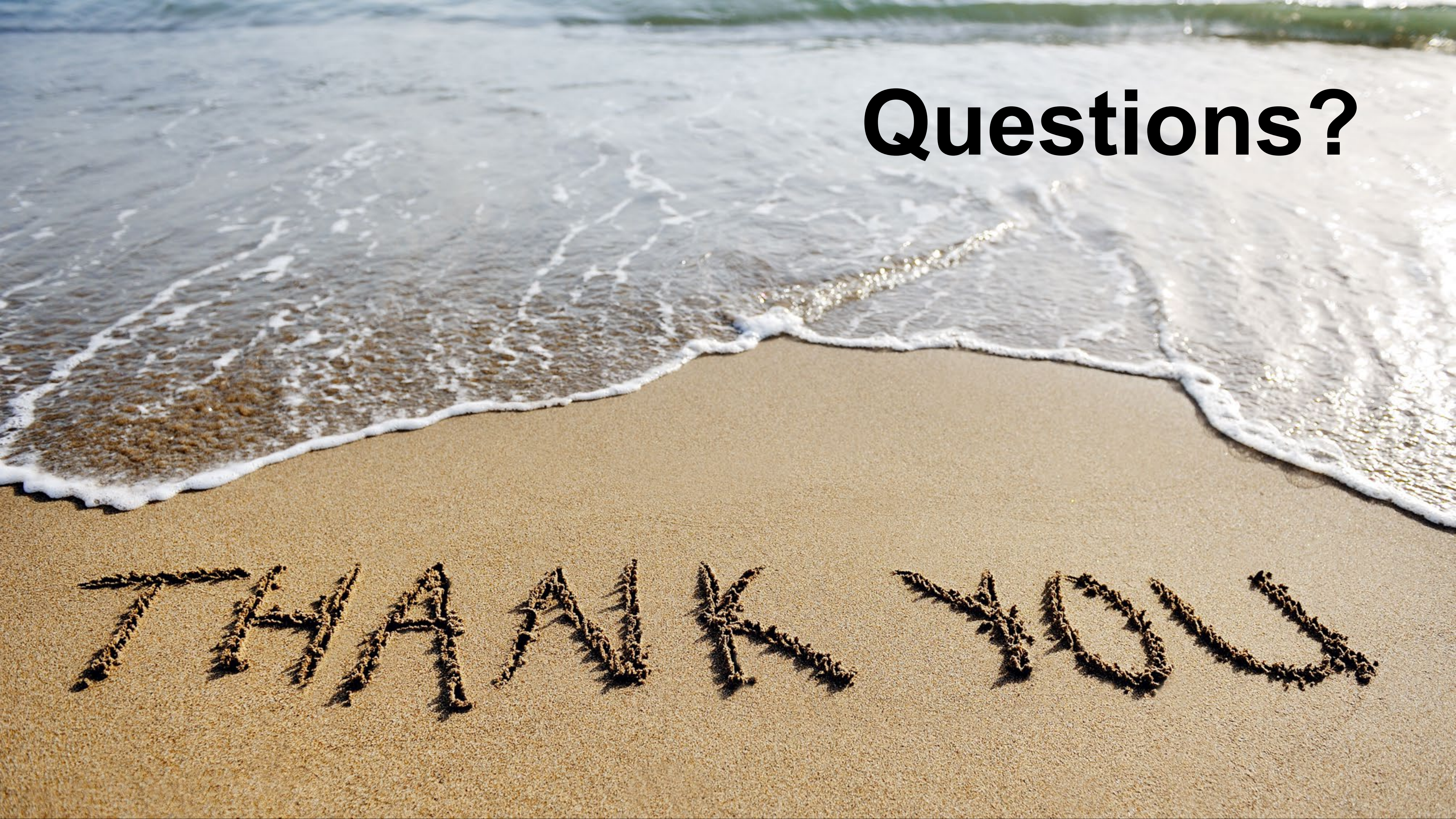
PODCAST

- Interview of Manager Marko Setälä from Nokia
  - Experience from many globally distributed projects
  - Many of them very large, > 1000 persons
- Interviewer: Casper Lassenius

ESSAY – GLOBAL SOFTWARE ENGINEERING

- Case: You are working as the manager for software development at a small web-software development company that is growing fast. You have now 20 persons working in development and you count that you would need to hire at least 20 more during the next year. You have now developed your product using Scrum for about one year. All your current customers are from Finland, but your company is planning to start marketing the product internationally. Your CEO has a new idea: Instead of hiring more developers from Finland he thinks the company could make much more money by hiring a subcontractor that has developers in India and Poland, as the salaries are lower in these countries than in Finland. He asks your opinion on whether it would make sense to hire this kind of a subcontractor and whether you would prefer developers from India or Poland.
- Prepare a 700-word essay style memo outlining your thoughts on the matter, especially pros and cons on whether to hire a subcontractor or not, and if hired, which country (India or Poland) you would prefer and why. Finally, give a motivated recommendation to the CEO. For full points, try to stay within the 25% limit.
- As material use the attached article, lecture and podcast by Marko Setälä (and whatever other material you find useful):
- Length: max 700 words
- The use of referencing is NOT needed in this assignment!

Questions?



LUT
University