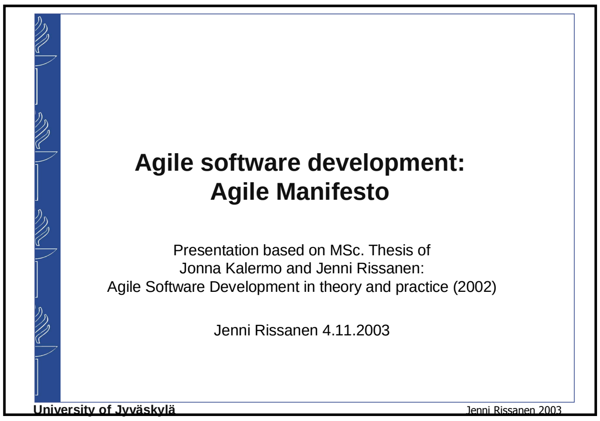
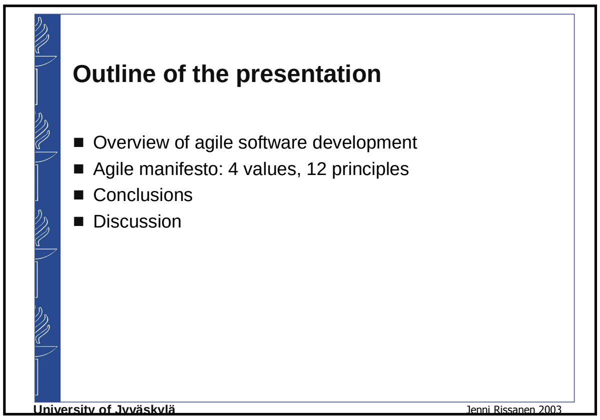


Agile software development:
Agile Manifesto

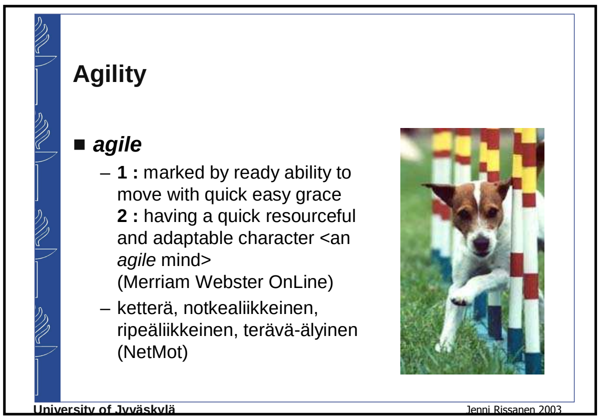
Presentation based on MSc. Thesis of
Jonna Kalermo and Jenni Rissanen:
Agile Software Development in theory and practice (2002)

Jenni Rissanen 4.11.2003



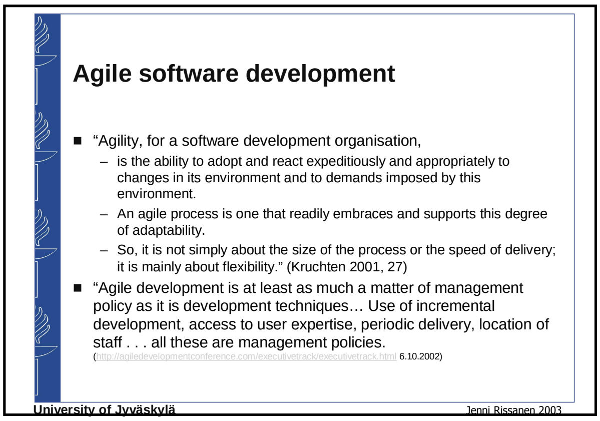
Outline of the presentation

- Overview of agile software development
- Agile manifesto: 4 values, 12 principles
- Conclusions
- Discussion



Agility

- ***agile***
  - **1 :** marked by ready ability to move with quick easy grace
    2 : having a quick resourceful and adaptable character <an *agile* mind>
    (Merriam Webster OnLine)
  - ketterä, notkealiikkeinen, ripeäliikkeinen, terävä-älyinen (NetMot)



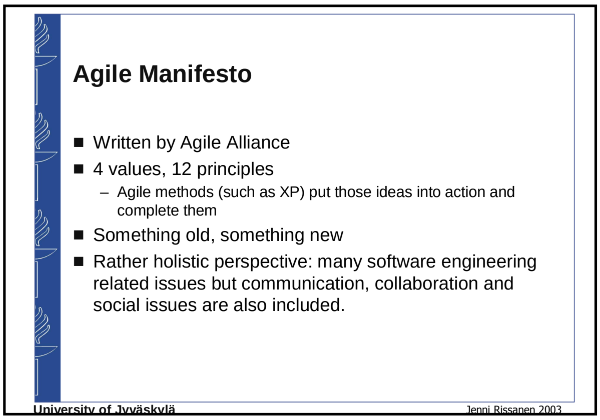
Agile software development

- “Agility, for a software development organisation,
  - is the ability to adopt and react expeditiously and appropriately to changes in its environment and to demands imposed by this environment.
  - An agile process is one that readily embraces and supports this degree of adaptability.
  - So, it is not simply about the size of the process or the speed of delivery; it is mainly about flexibility.” (Kruchten 2001, 27)
- “Agile development is at least as much a matter of management policy as it is development techniques… Use of incremental development, access to user expertise, periodic delivery, location of staff . . . all these are management policies.

(http://agiledevelopmentconference.com/executivetrack/executivetrack.html 6.10.2002)

University of Jyväskylä

Jenni Rissanen 2003

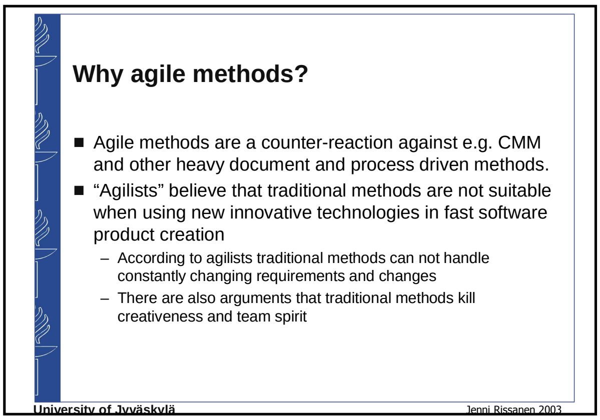


Agile Manifesto

- Written by Agile Alliance
- 4 values, 12 principles
  - Agile methods (such as XP) put those ideas into action and complete them
- Something old, something new
- Rather holistic perspective: many software engineering related issues but communication, collaboration and social issues are also included.

University of Jyväskylä

Jenni Rissanen 2003

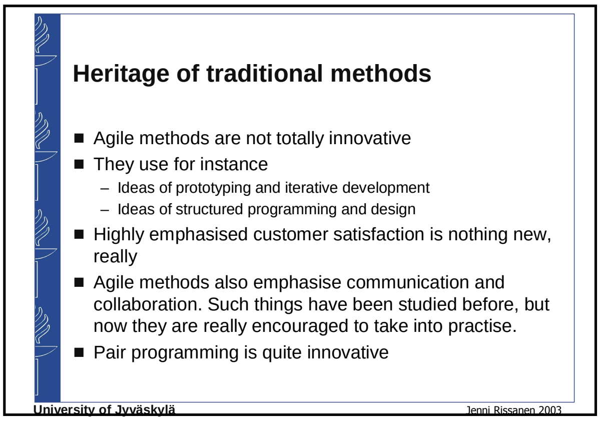


Why agile methods?

- Agile methods are a counter-reaction against e.g. CMM and other heavy document and process driven methods.
- “Agilists” believe that traditional methods are not suitable when using new innovative technologies in fast software product creation
  - According to agilists traditional methods can not handle constantly changing requirements and changes
  - There are also arguments that traditional methods kill creativeness and team spirit

University of Jyväskylä

Jenni Rissanen 2003

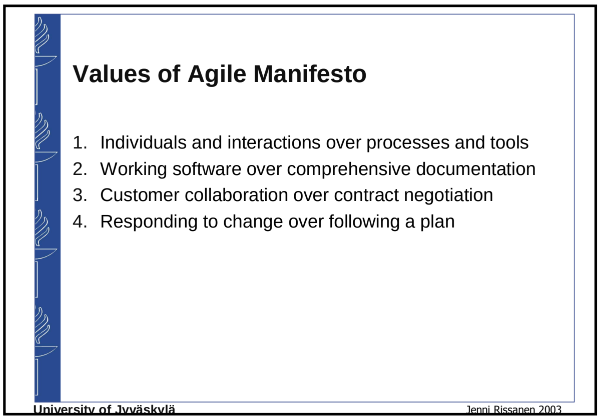


Heritage of traditional methods

- Agile methods are not totally innovative
- They use for instance
  - Ideas of prototyping and iterative development
  - Ideas of structured programming and design
- Highly emphasised customer satisfaction is nothing new, really
- Agile methods also emphasise communication and collaboration. Such things have been studied before, but now they are really encouraged to take into practise.
- Pair programming is quite innovative

University of Jyväskylä

Jenni Rissanen 2003

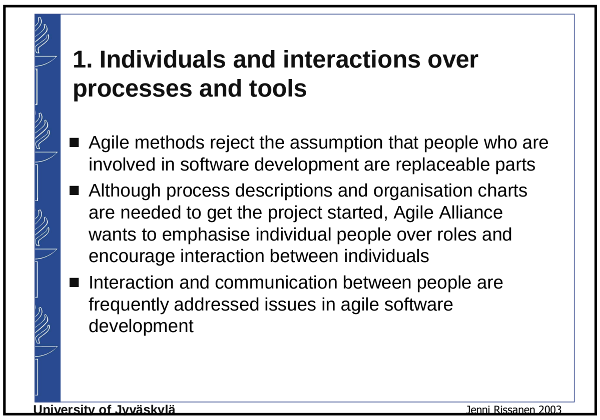


Values of Agile Manifesto

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

University of Jyväskylä

Jenni Rissanen 2003

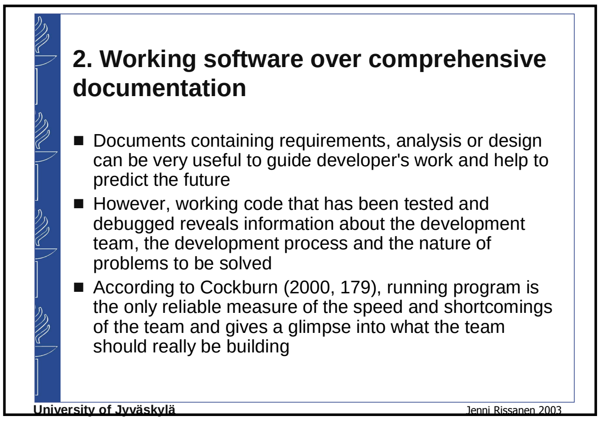


1. Individuals and interactions over processes and tools

- Agile methods reject the assumption that people who are involved in software development are replaceable parts
- Although process descriptions and organisation charts are needed to get the project started, Agile Alliance wants to emphasise individual people over roles and encourage interaction between individuals
- Interaction and communication between people are frequently addressed issues in agile software development

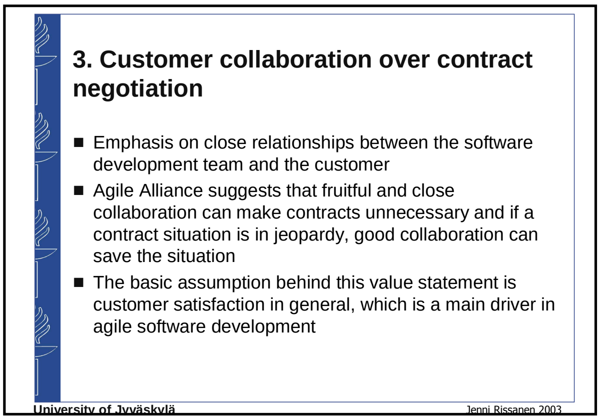
University of Jyväskylä

Jenni Rissanen 2003



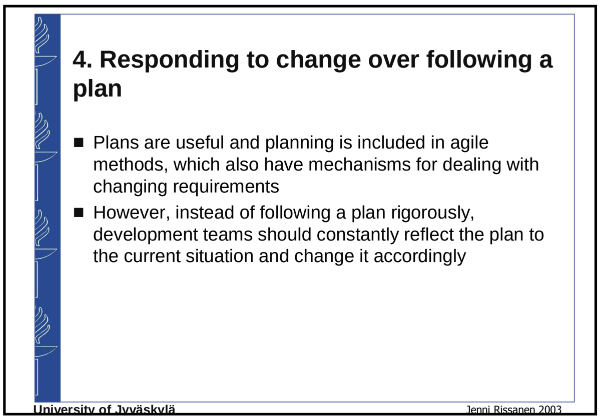
2. Working software over comprehensive documentation

- Documents containing requirements, analysis or design can be very useful to guide developer's work and help to predict the future
- However, working code that has been tested and debugged reveals information about the development team, the development process and the nature of problems to be solved
- According to Cockburn (2000, 179), running program is the only reliable measure of the speed and shortcomings of the team and gives a glimpse into what the team should really be building



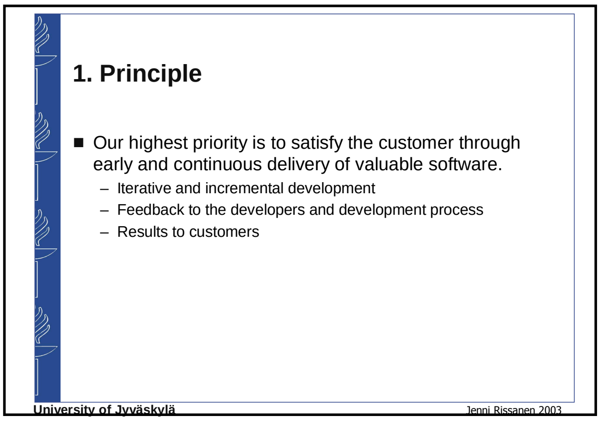
3. Customer collaboration over contract negotiation

- Emphasis on close relationships between the software development team and the customer
- Agile Alliance suggests that fruitful and close collaboration can make contracts unnecessary and if a contract situation is in jeopardy, good collaboration can save the situation
- The basic assumption behind this value statement is customer satisfaction in general, which is a main driver in agile software development



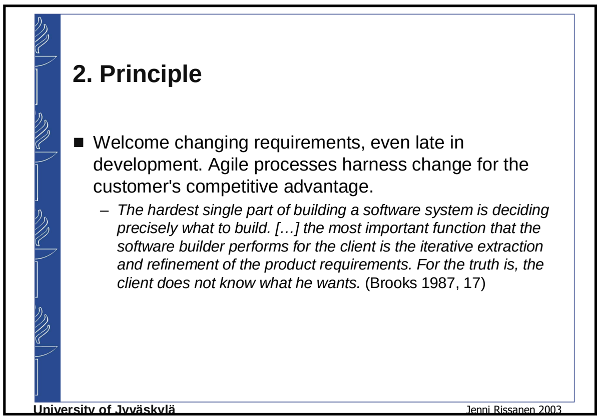
4. Responding to change over following a plan

- Plans are useful and planning is included in agile methods, which also have mechanisms for dealing with changing requirements
- However, instead of following a plan rigorously, development teams should constantly reflect the plan to the current situation and change it accordingly



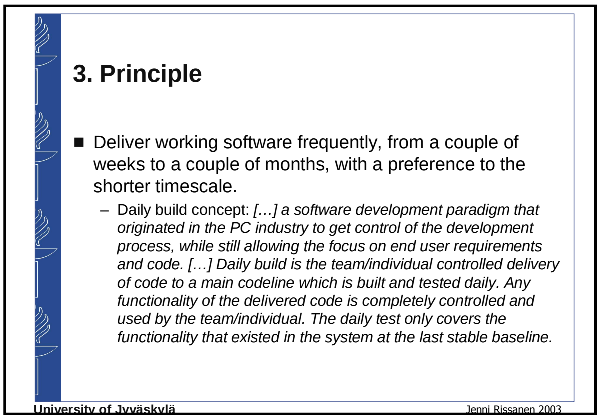
1. Principle

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
  - Iterative and incremental development
  - Feedback to the developers and development process
  - Results to customers



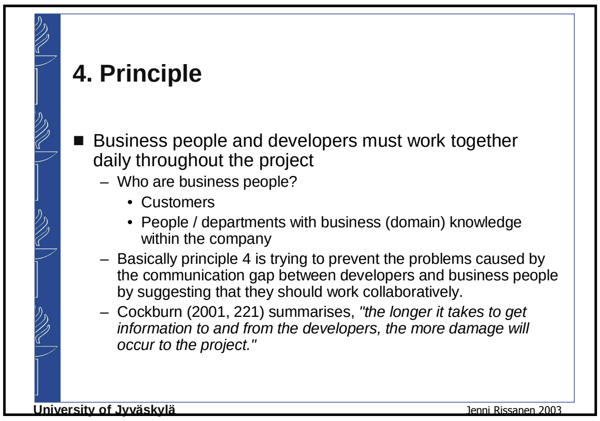
2. Principle

- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
  - *The hardest single part of building a software system is deciding precisely what to build. […] the most important function that the software builder performs for the client is the iterative extraction and refinement of the product requirements. For the truth is, the client does not know what he wants.* (Brooks 1987, 17)



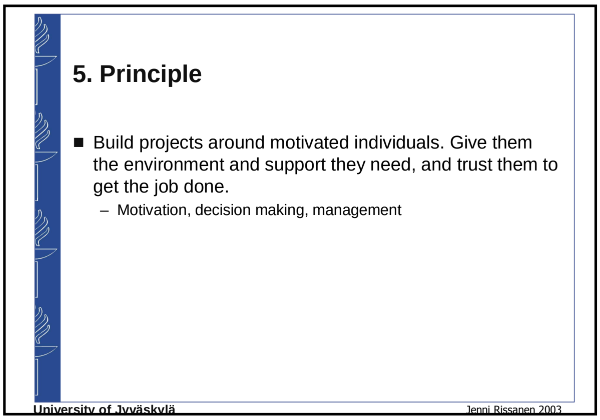
3. Principle

- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
  - Daily build concept: *[…] a software development paradigm that originated in the PC industry to get control of the development process, while still allowing the focus on end user requirements and code. […] Daily build is the team/individual controlled delivery of code to a main codeline which is built and tested daily. Any functionality of the delivered code is completely controlled and used by the team/individual. The daily test only covers the functionality that existed in the system at the last stable baseline.*



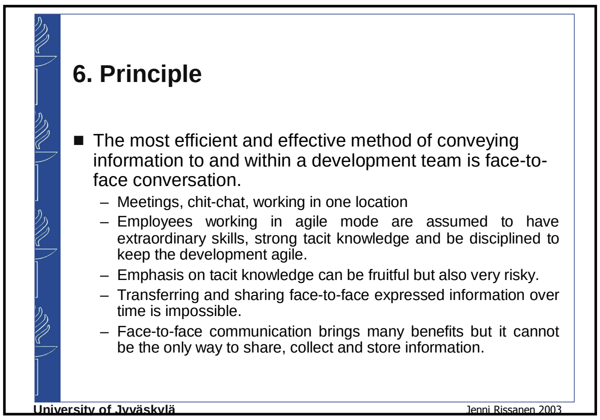
4. Principle

- Business people and developers must work together daily throughout the project
  - Who are business people?
    - Customers
    - People / departments with business (domain) knowledge within the company
  - Basically principle 4 is trying to prevent the problems caused by the communication gap between developers and business people by suggesting that they should work collaboratively.
  - Cockburn (2001, 221) summarises, *"the longer it takes to get information to and from the developers, the more damage will occur to the project."*



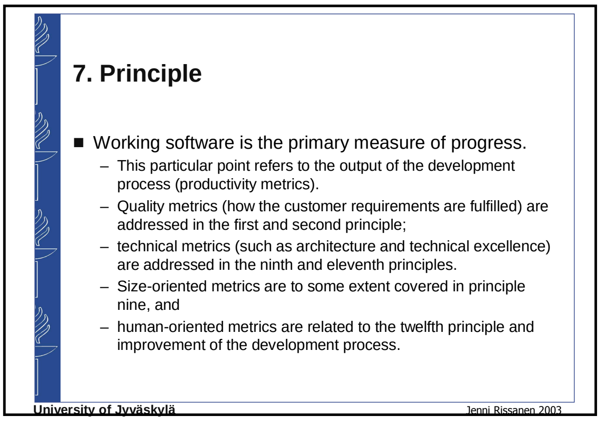
5. Principle

- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
  - Motivation, decision making, management



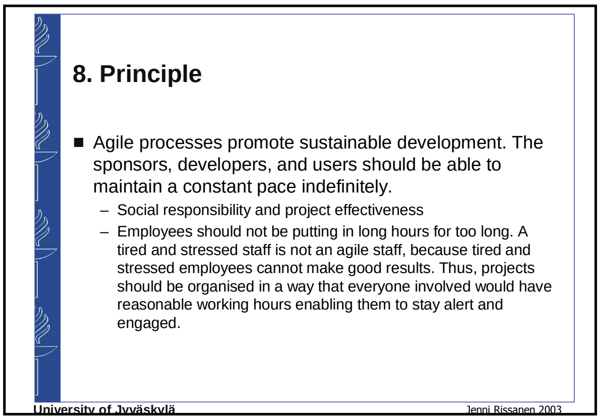
6. Principle

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
  - Meetings, chit-chat, working in one location
  - Employees working in agile mode are assumed to have extraordinary skills, strong tacit knowledge and be disciplined to keep the development agile.
  - Emphasis on tacit knowledge can be fruitful but also very risky.
  - Transferring and sharing face-to-face expressed information over time is impossible.
  - Face-to-face communication brings many benefits but it cannot be the only way to share, collect and store information.



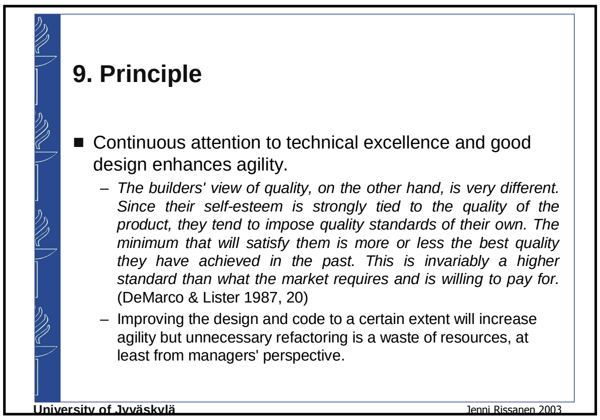
7. Principle

- Working software is the primary measure of progress.
  - This particular point refers to the output of the development process (productivity metrics).
  - Quality metrics (how the customer requirements are fulfilled) are addressed in the first and second principle;
  - technical metrics (such as architecture and technical excellence) are addressed in the ninth and eleventh principles.
  - Size-oriented metrics are to some extent covered in principle nine, and
  - human-oriented metrics are related to the twelfth principle and improvement of the development process.



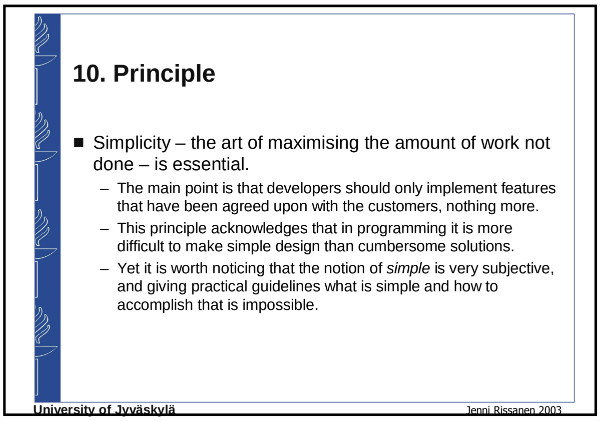
8. Principle

- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
  - Social responsibility and project effectiveness
  - Employees should not be putting in long hours for too long. A tired and stressed staff is not an agile staff, because tired and stressed employees cannot make good results. Thus, projects should be organised in a way that everyone involved would have reasonable working hours enabling them to stay alert and engaged.



9. Principle

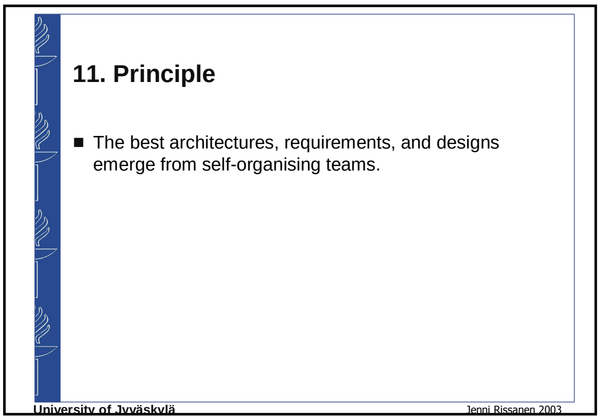
- Continuous attention to technical excellence and good design enhances agility.
  - *The builders' view of quality, on the other hand, is very different. Since their self-esteem is strongly tied to the quality of the product, they tend to impose quality standards of their own. The minimum that will satisfy them is more or less the best quality they have achieved in the past. This is invariably a higher standard than what the market requires and is willing to pay for.* (DeMarco & Lister 1987, 20)
  - Improving the design and code to a certain extent will increase agility but unnecessary refactoring is a waste of resources, at least from managers' perspective.



10. Principle

- Simplicity – the art of maximising the amount of work not done – is essential.
  - The main point is that developers should only implement features that have been agreed upon with the customers, nothing more.
  - This principle acknowledges that in programming it is more difficult to make simple design than cumbersome solutions.
  - Yet it is worth noticing that the notion of *simple* is very subjective, and giving practical guidelines what is simple and how to accomplish that is impossible.

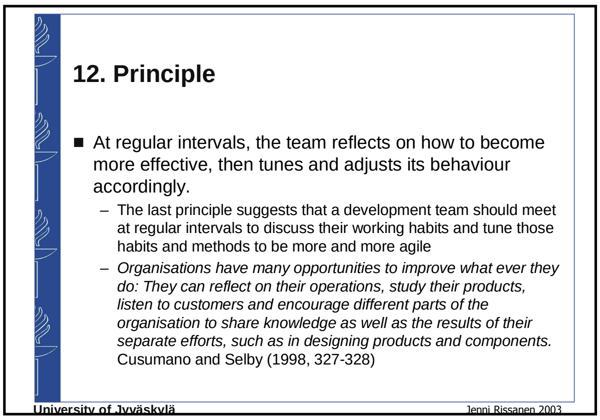
University of Jyväskylä Jenni Rissanen 2003



11. Principle

- The best architectures, requirements, and designs emerge from self-organising teams.

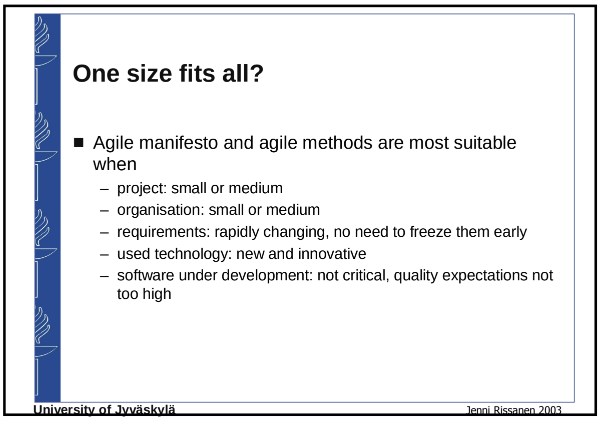
University of Jyväskylä Jenni Rissanen 2003



12. Principle

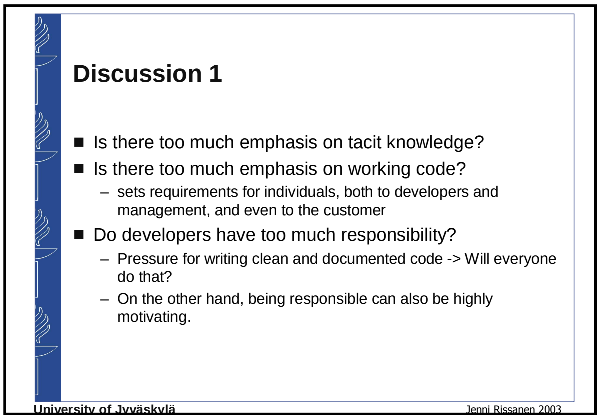
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.
  - The last principle suggests that a development team should meet at regular intervals to discuss their working habits and tune those habits and methods to be more and more agile
  - *Organisations have many opportunities to improve what ever they do: They can reflect on their operations, study their products, listen to customers and encourage different parts of the organisation to share knowledge as well as the results of their separate efforts, such as in designing products and components.* Cusumano and Selby (1998, 327-328)

University of Jyväskylä Jenni Rissanen 2003



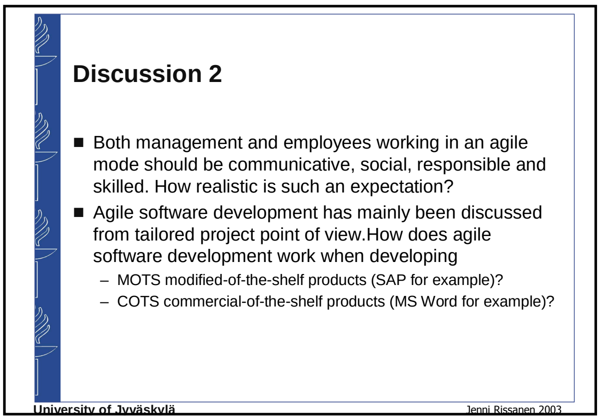
One size fits all?

- Agile manifesto and agile methods are most suitable when
  - project: small or medium
  - organisation: small or medium
  - requirements: rapidly changing, no need to freeze them early
  - used technology: new and innovative
  - software under development: not critical, quality expectations not too high



Discussion 1

- Is there too much emphasis on tacit knowledge?
- Is there too much emphasis on working code?
  - sets requirements for individuals, both to developers and management, and even to the customer
- Do developers have too much responsibility?
  - Pressure for writing clean and documented code -> Will everyone do that?
  - On the other hand, being responsible can also be highly motivating.



Discussion 2

- Both management and employees working in an agile mode should be communicative, social, responsible and skilled. How realistic is such an expectation?
- Agile software development has mainly been discussed from tailored project point of view.How does agile software development work when developing
  - MOTS modified-of-the-shelf products (SAP for example)?
  - COTS commercial-of-the-shelf products (MS Word for example)?