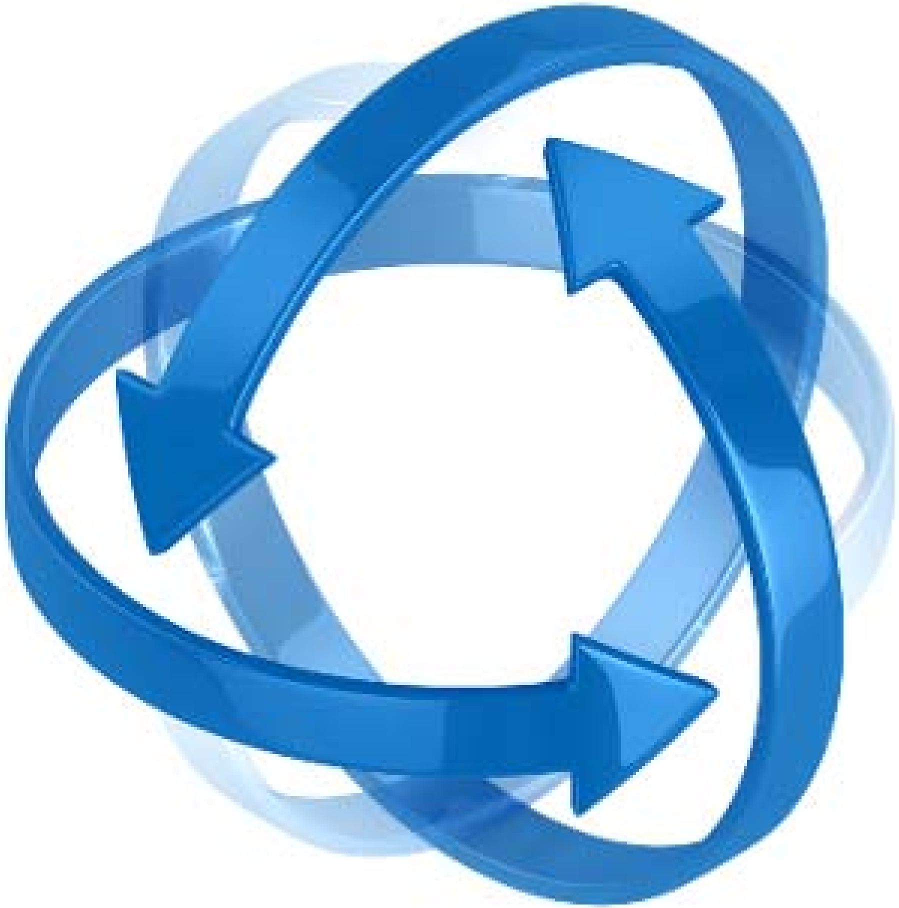
Chapter 3 – Agile Software Development



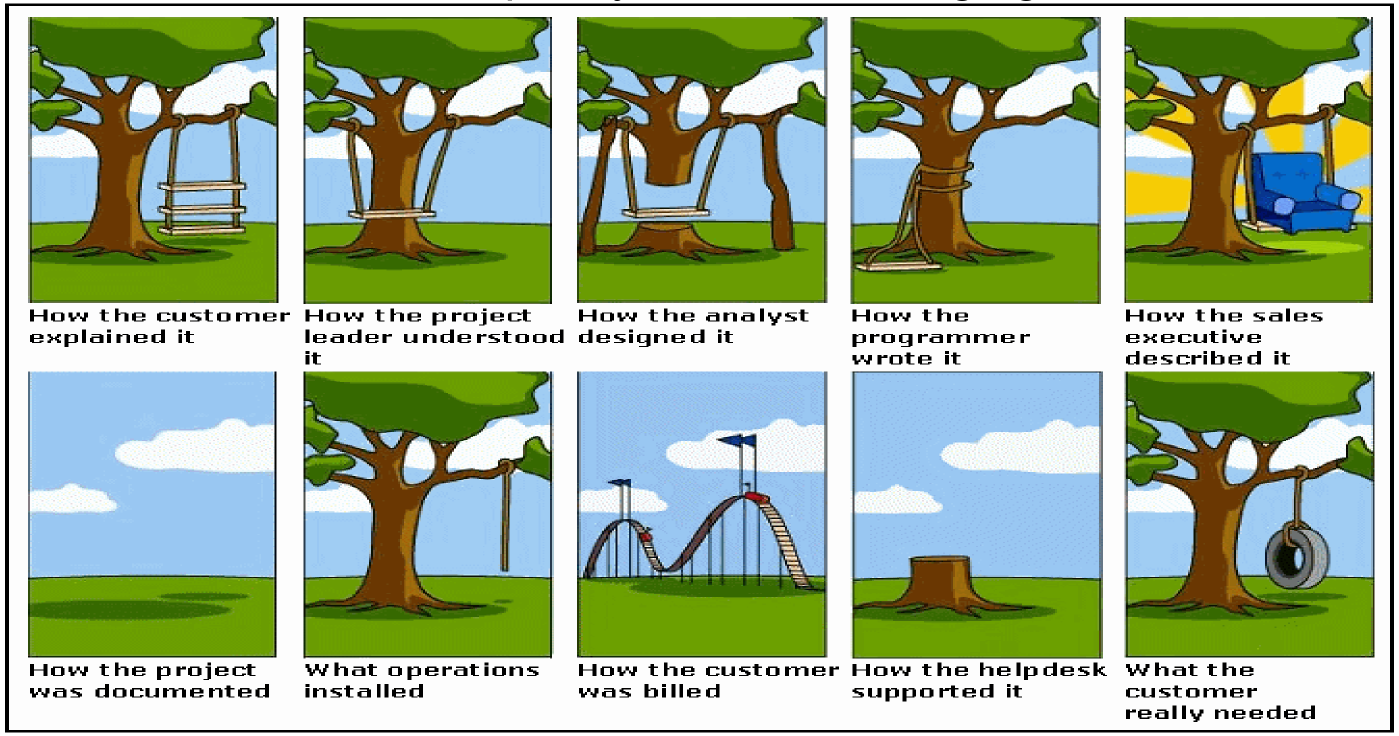
Lecture 1

Topics covered

- Agile methods
- Plan-driven and agile development
- Extreme programming
- Agile project management
- Scaling agile methods

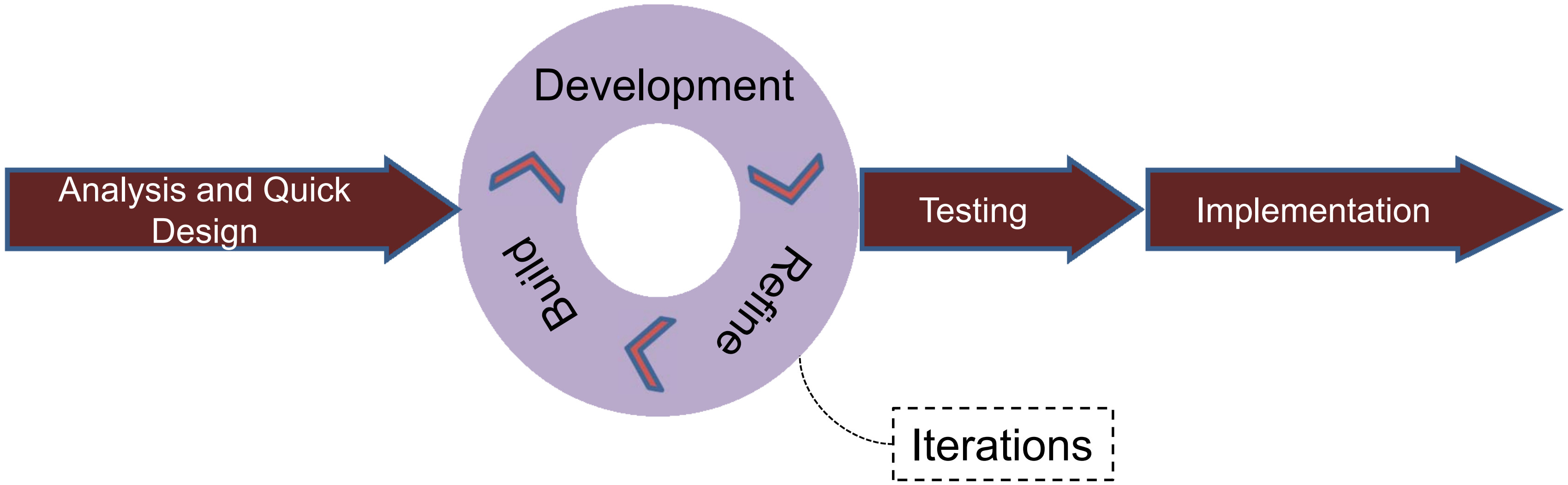
Rapid software development

- Rapid development and delivery: A basic requirement
  - Businesses operate in a fast –changing requirement. It is practically impossible to produce a set of stable software requirements
  - Software has to evolve quickly to reflect changing business needs


How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
How the sales executive described it
How the project was documented
What operations installed
How the customer was billed
How the helpdesk supported it
What the customer really needed

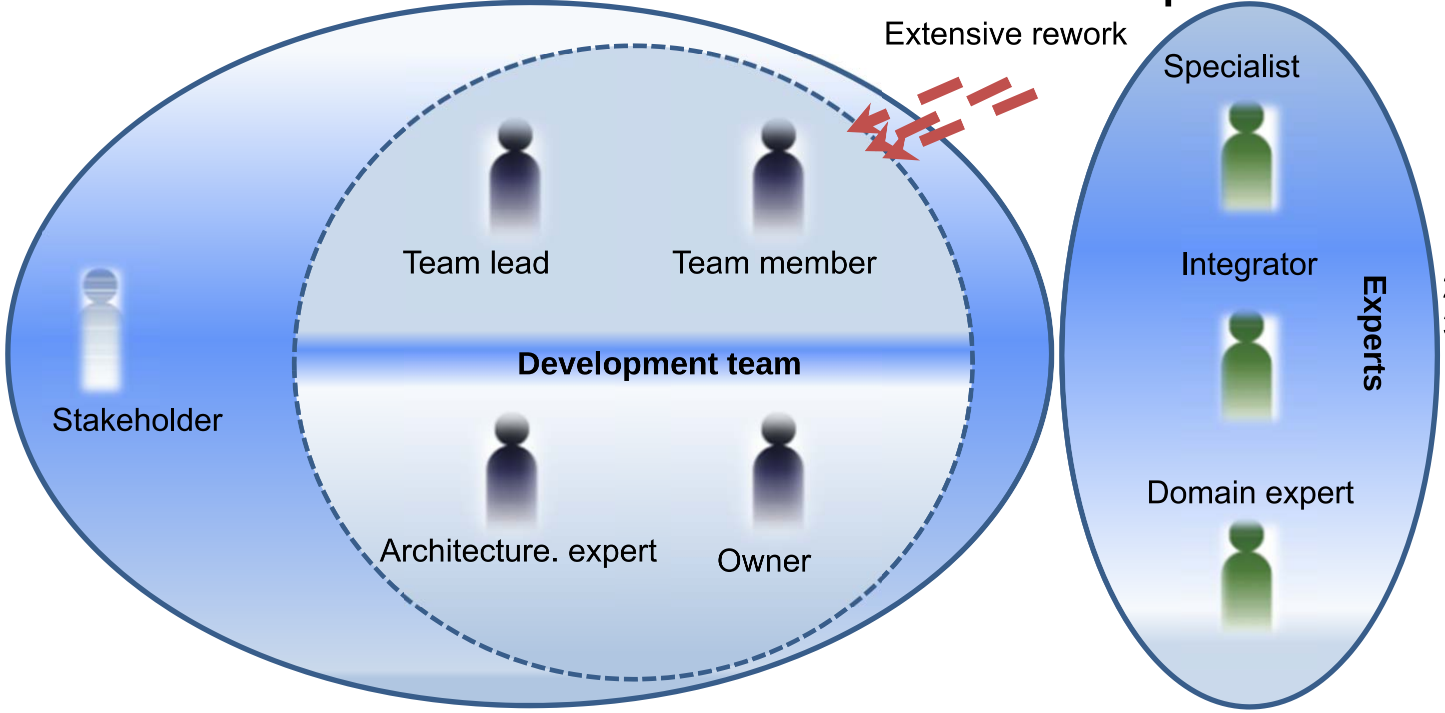
Rapid software development

- Rapid software development
  - Specification, design and implementation are inter-leaved
  - System is developed as a series of versions with stakeholders involved in version evaluation
  - User interfaces are often developed using an IDE and graphical toolset.



Agile methods

- ✧ Dissatisfaction with legacy methods of 1980s, 1990 led to agile methods
- ✧ Aim: To reduce overheads in software process



Agile Methods Focus

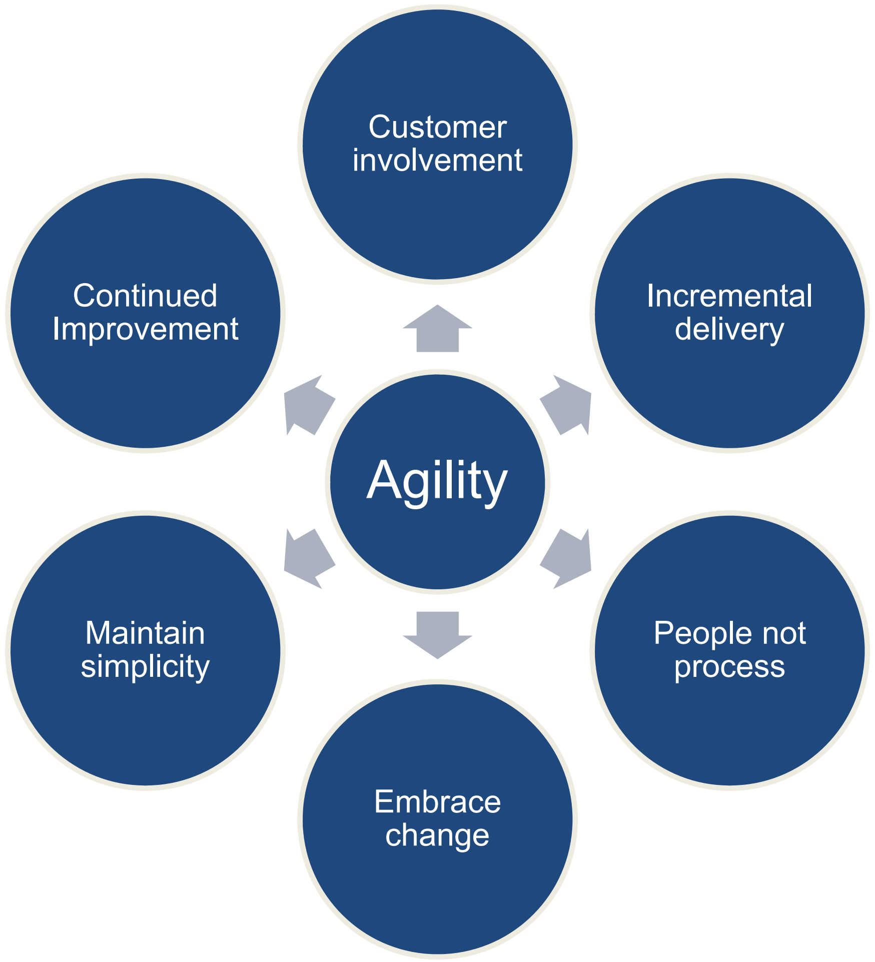
1. Code rather than design
2. Iterative
3. Evolution

Agile manifesto

Agile method preferences

Agile Methods	**Legacy Methods**
Individual and interaction	Process and tools
Working software	Comprehensive documents
Customer collaboration	Contract negotiation
Responding to change	Following a plan

Principles of agile methods



Principles of agile methods: Explained

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

I need agile methodology: What is It?

Agile method applicability

Agile methods can be:

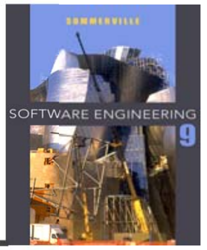
- People-centric: way to create innovative solutions
- Product-centric: alternative to documents/process
- Market-centric: model to maximize business value

Two types of development with Agile Methods:

Product development	Custom system development
Small size software	Customer fully involved in the development process
Medium size software	Not a lot of external rules and regulations

http://semanticcommunity.info/AOL_Government/ACT-IAC_Agile_Development

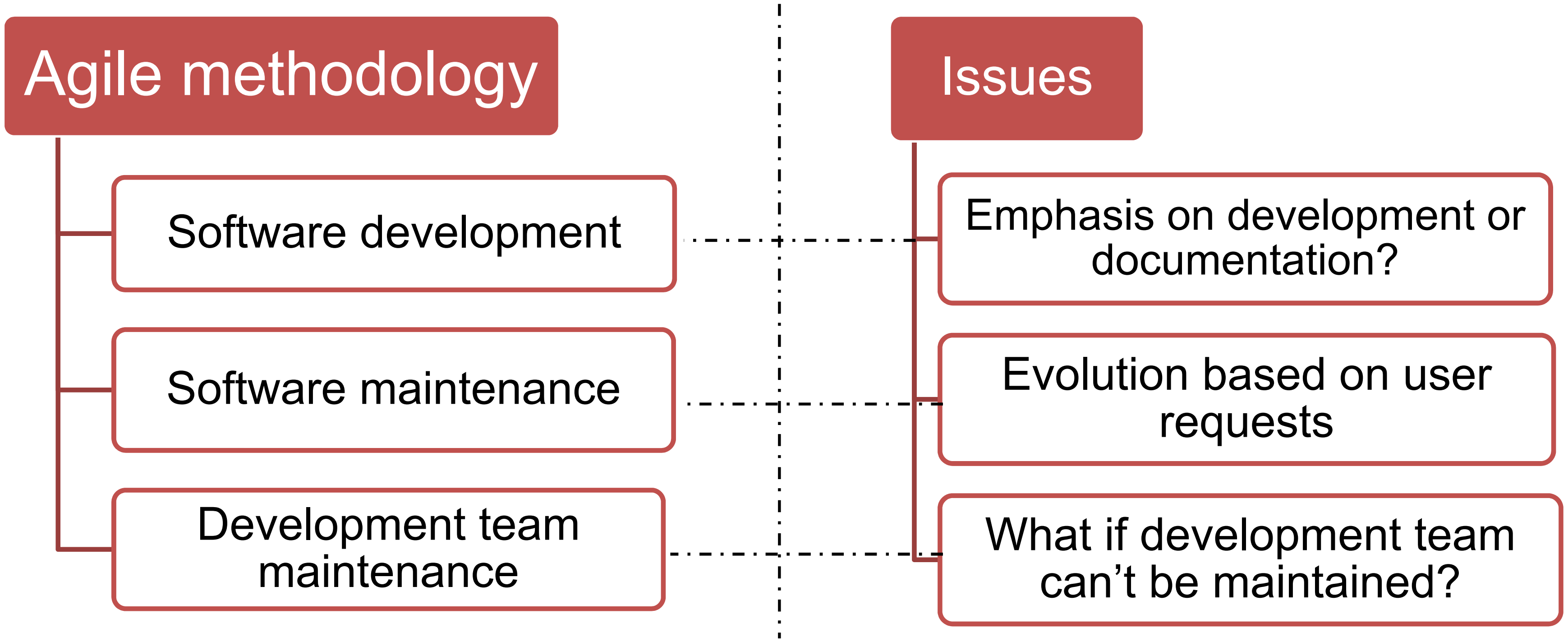
Problems with agile methods



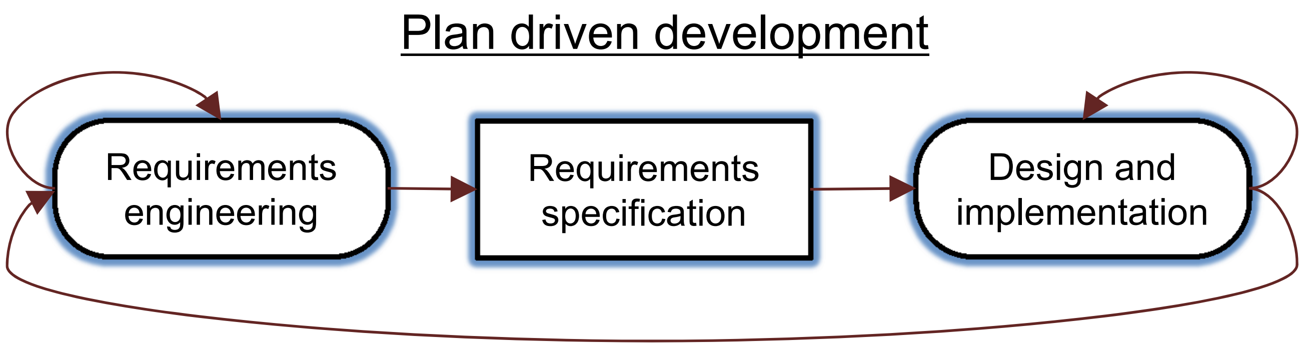
- Difficulty in keeping the interest of involved customers
- Unsuited team members with characteristics of agility
- Multiple stakeholders(difficulty in prioritizing updates)
- Simplicity costs extra work
- Contracts may be a problem

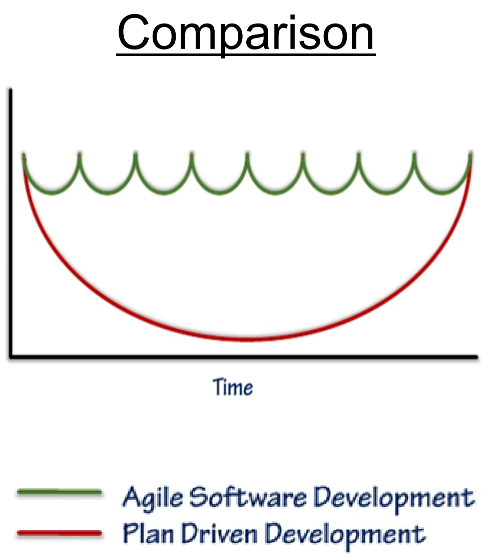
Agile methods and software maintenance

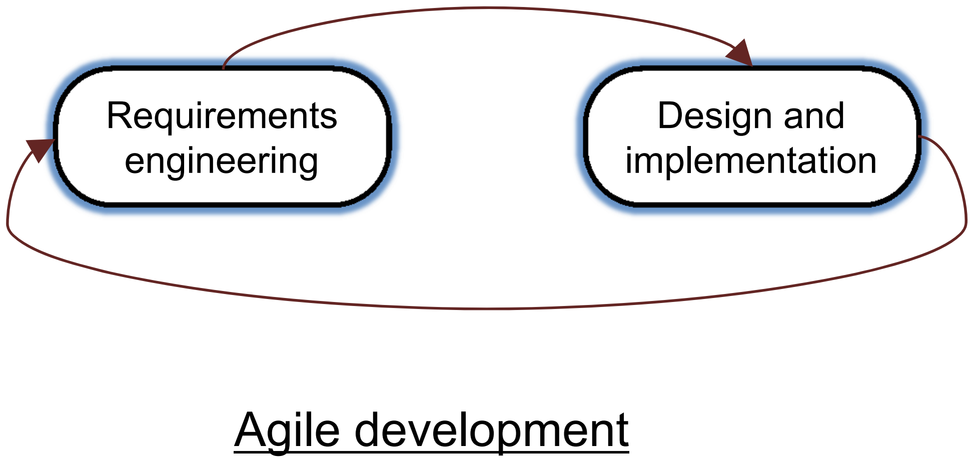
- Most organizations spend more on maintaining existing software than they do on new software development. So, if agile methods are to be successful, they have to support maintenance as well as original development


Agile methodology
Software development
Software maintenance
Development team maintenance
Issues
Emphasis on development or documentation?
Evolution based on user requests
What if development team can't be maintained?

Plan-driven and agile development specification



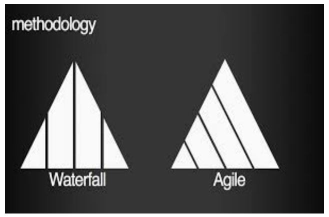




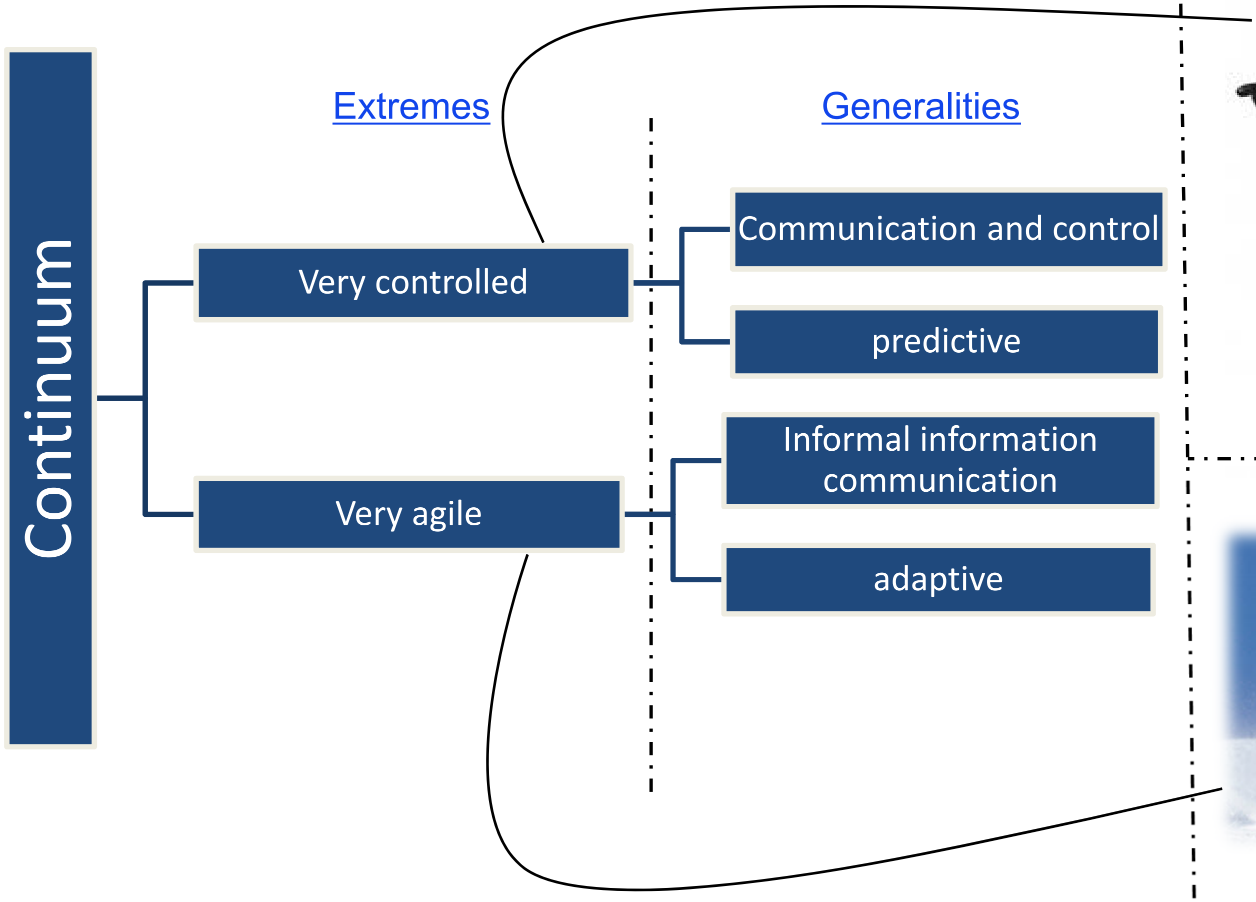
Plan-driven vs. Agile methods

- Most projects include elements of plan-driven and agile processes. Deciding on the balance depends on:
  - Detailed specification and design before moving to implementation? If so, you probably need to use a plan-driven approach.
  - Is an incremental delivery strategy and feedback from customers, realistic? If so, consider using agile methods.
  - How large is the system that is being developed?
    - Small co-located team who can communicate informally: Agile method
    - Large systems that require larger development teams: Plan-driven approach

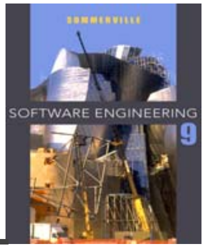
From waterfall to agile

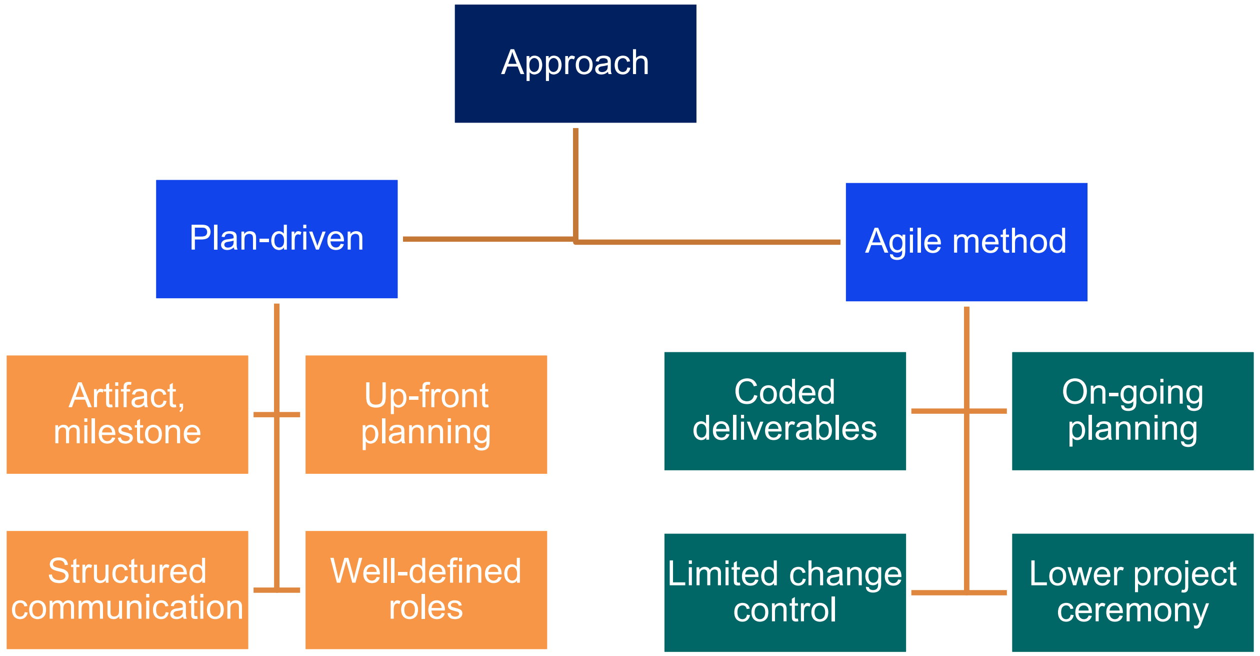


Plan-driven vs. Agile methods



Plan-driven vs. Agile methods





Technical, human, organizational issues

- Most projects include elements of plan-driven and agile processes. Deciding on the balance depends on:
  - Is it important to have a very detailed specification and design before moving to implementation? If so, you probably need to use a plan-driven approach.
  - Is an incremental delivery strategy, where you deliver the software to customers and get rapid feedback from them, realistic? If so, consider using agile methods.
  - How large is the system that is being developed? Agile methods are most effective when the system can be developed with a small co-located team who can communicate informally. This may not be possible for large systems that require larger development teams so a plan-driven approach may have to be used.

Technical, human, organizational issues

Issues

What type of system is being developed?

Plan-driven approaches may be required for systems that require a lot of analysis before implementation

What is expected lifetime of the system?

Long-lifetime systems require more design documentation to communicate the original intentions of the system developers to the support team

What technologies are available and how is team organized?

Agile methods rely on good tools.

If the development team is distributed or if part of the development is being outsourced

Technical, human, organizational issues

Issues

Are there cultural or organizational issues that may affect the system development?

Traditional engineering organizations have a culture of plan-based development, as this is the norm in engineering

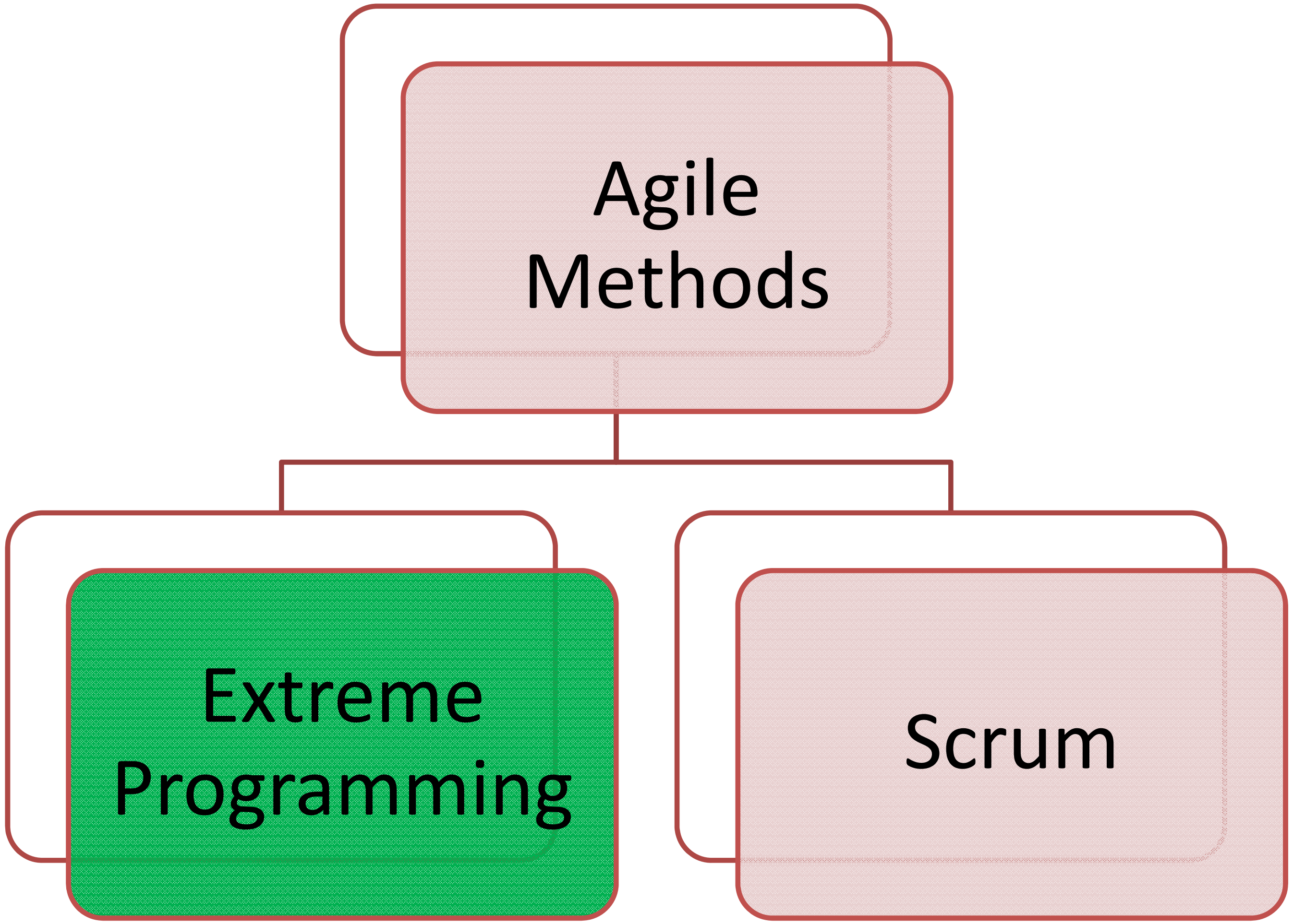
How good are the designers and programmers in the development team?

It is sometimes argued that agile methods require higher skill levels than plan-based approaches in which programmers simply translate a detailed design into code.

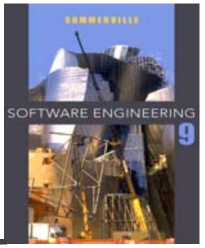
Is the system subject to external regulation?

If a system has to be approved by an external regulator (e.g. the FAA approve software that is critical to the operation of an aircraft) then you will probably be required to produce detailed documentation as part of the system safety case.

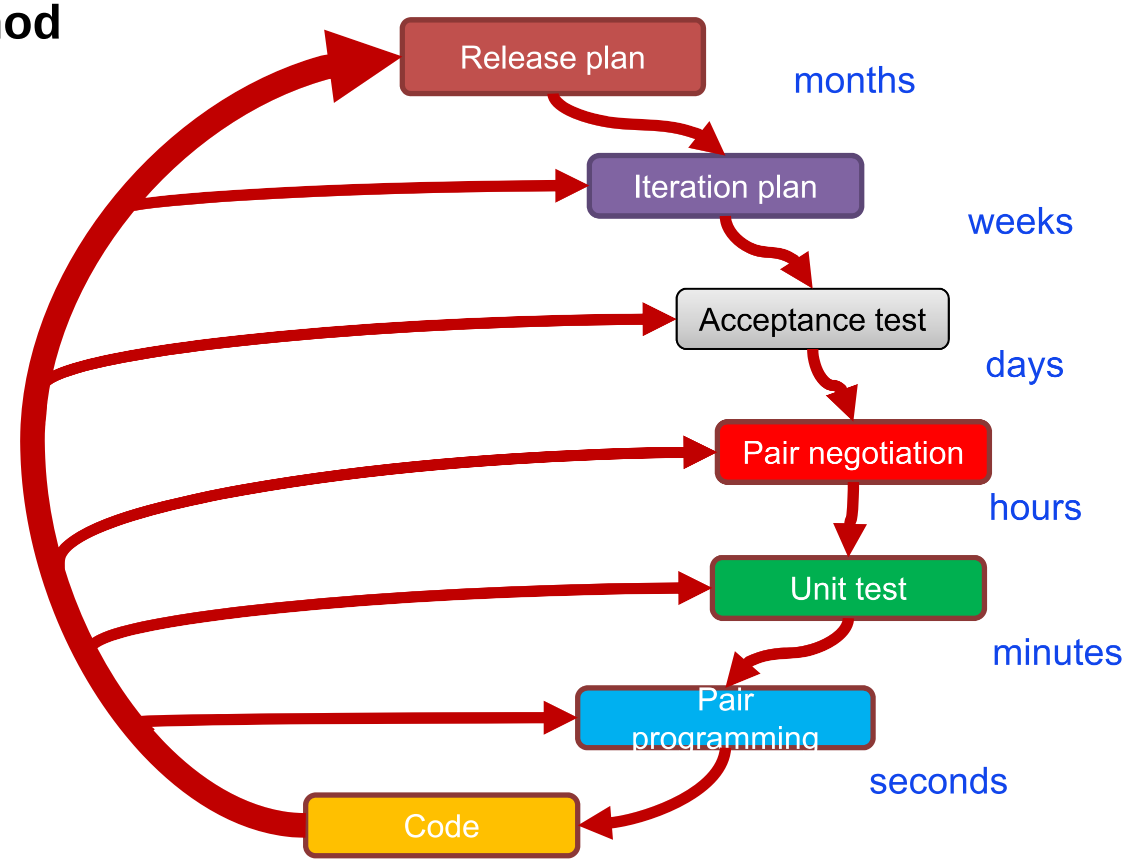
Agile Methods in this lecture



Extreme programming



Perhaps the best-known and most widely used agile method

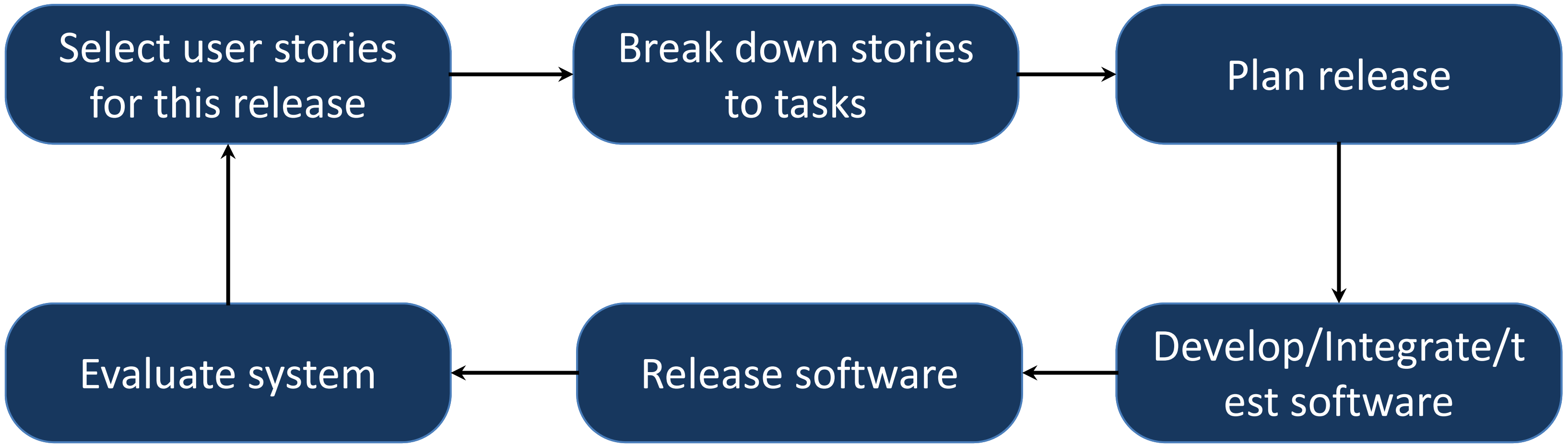


Ref: http://petercodes.wordpress.com/2013/11/09/personal-extreme-programming-part-2-why-agile-and-why-xp/

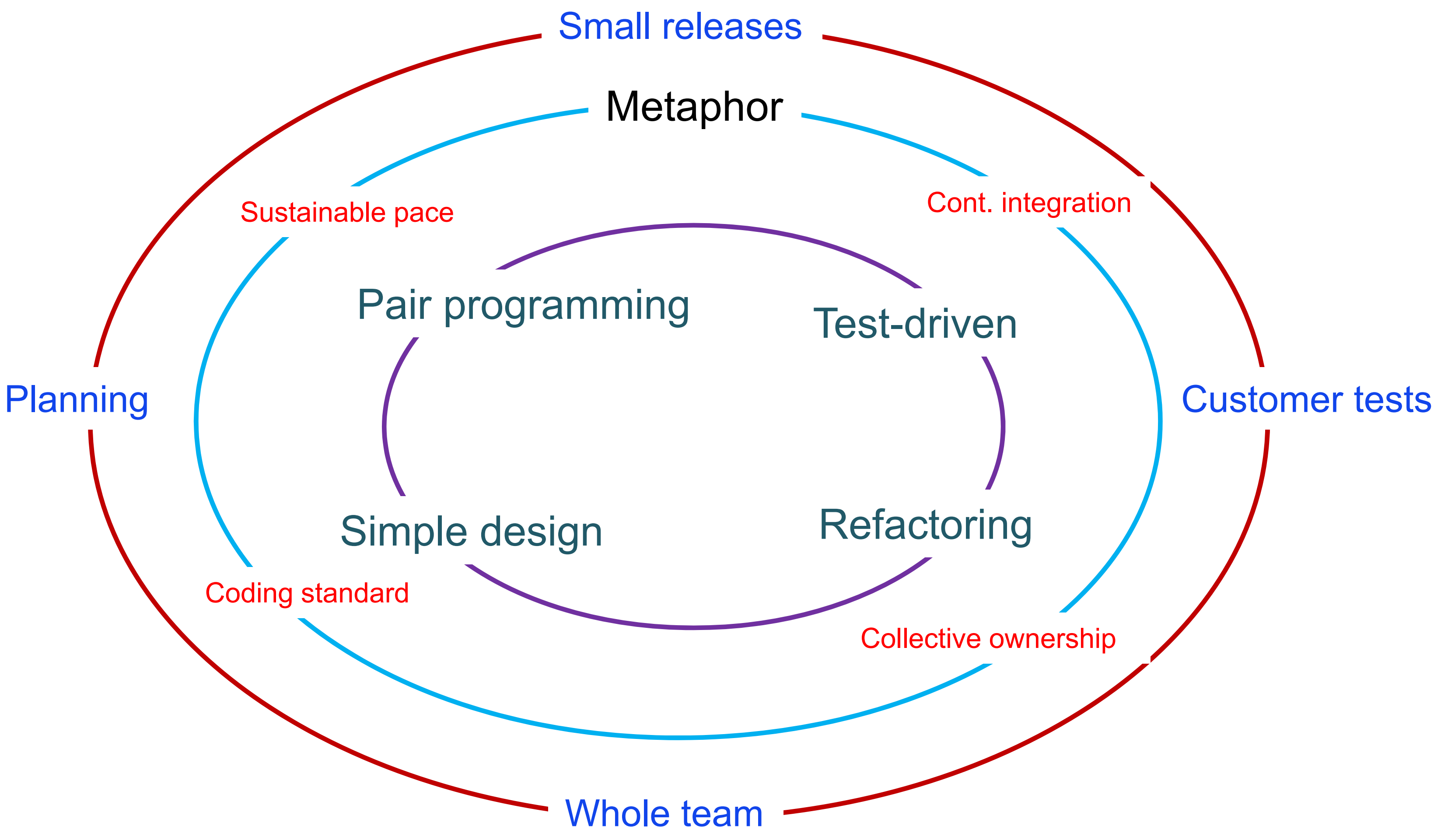
XP and agile principles

- Incremental development is supported through small, frequent system releases.
- Customer involvement means full-time customer engagement with the team.
- People not process through pair programming.
- Change supported through regular system releases.
- Maintaining simplicity through constant refactoring of code.

XP release cycle


Select user stories for this release
Break down stories to tasks
Plan release
Develop/Integrate/test software
Release software
Evaluate system

Extreme programming practices



Ref: http://xprogramming.com/what-is-extreme-programming/

Extreme programming practices (a) Detailed

Principle or practice	Description
Incremental planning	Requirements are recorded on story cards and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

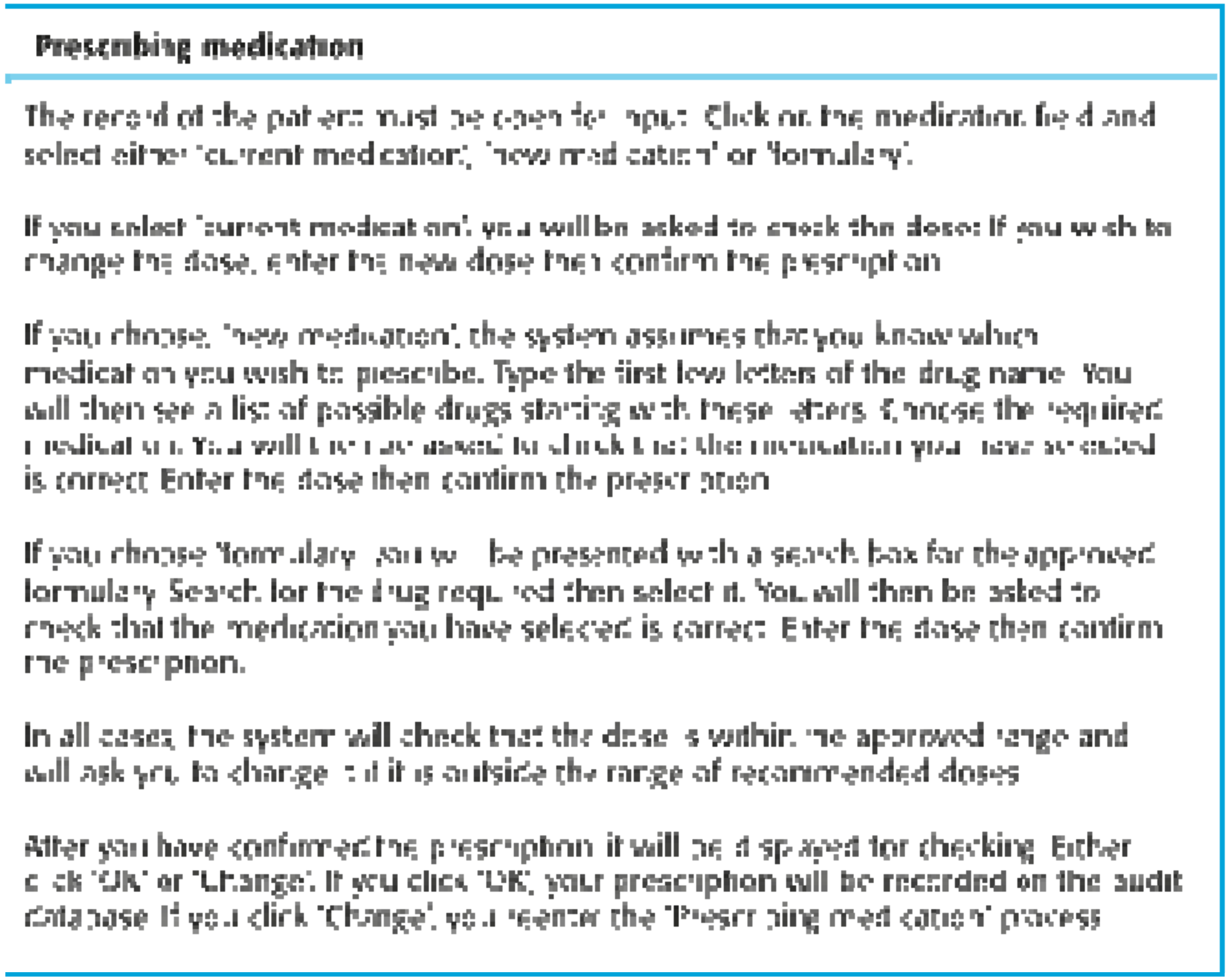
Extreme programming practices (b) Detailed

Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

Requirements scenarios

- In XP, a customer or user is part of the XP team and is responsible for making decisions on requirements.
- User requirements are expressed as scenarios or user stories.
- These are written on cards and the development team break them down into implementation tasks. These tasks are the basis of schedule and cost estimates.
- The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.

A ‘prescribing medication’ story



Prescribing medication

The record of the patient must be open for input. Click on the medication field and select either 'current medication', 'new medication' or 'formulary'.

If you select 'current medication', you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

If you choose, 'new medication', the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

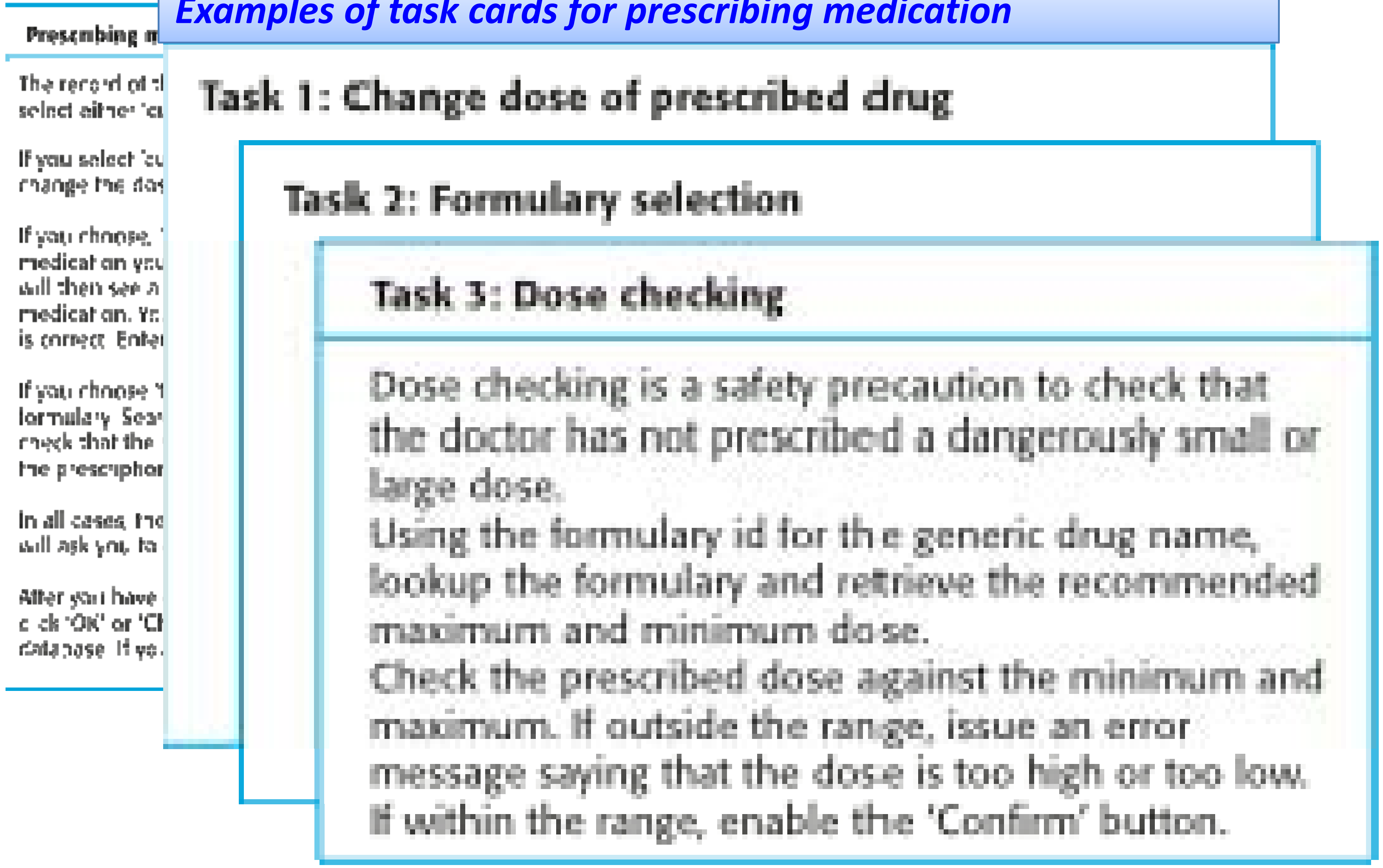
If you choose 'formulary', you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click 'OK' or 'Change'. If you click 'OK', your prescription will be recorded on the audit database. If you click 'Change', you reenter the 'Prescribing medication' process.

A ‘prescribing medication’ story

Examples of task cards for prescribing medication



Prescribing m

The record of t
select either 'cu

If you select 'cu
change the dos

If you choose '
medication you
will then see a
medication. Yo
is correct. Ente

If you choose '
formulary. Sea
check that the
the prescription

In all cases, the
will ask you to

After you have
click 'OK' or 'Ch
database. If yo

Task 1: Change dose of prescribed drug

Task 2: Formulary selection

Task 3: Dose checking

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.
Using the formulary id for the generic drug name, lookup the formulary and retrieve the recommended maximum and minimum dose.
Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low. If within the range, enable the 'Confirm' button.

XP and change

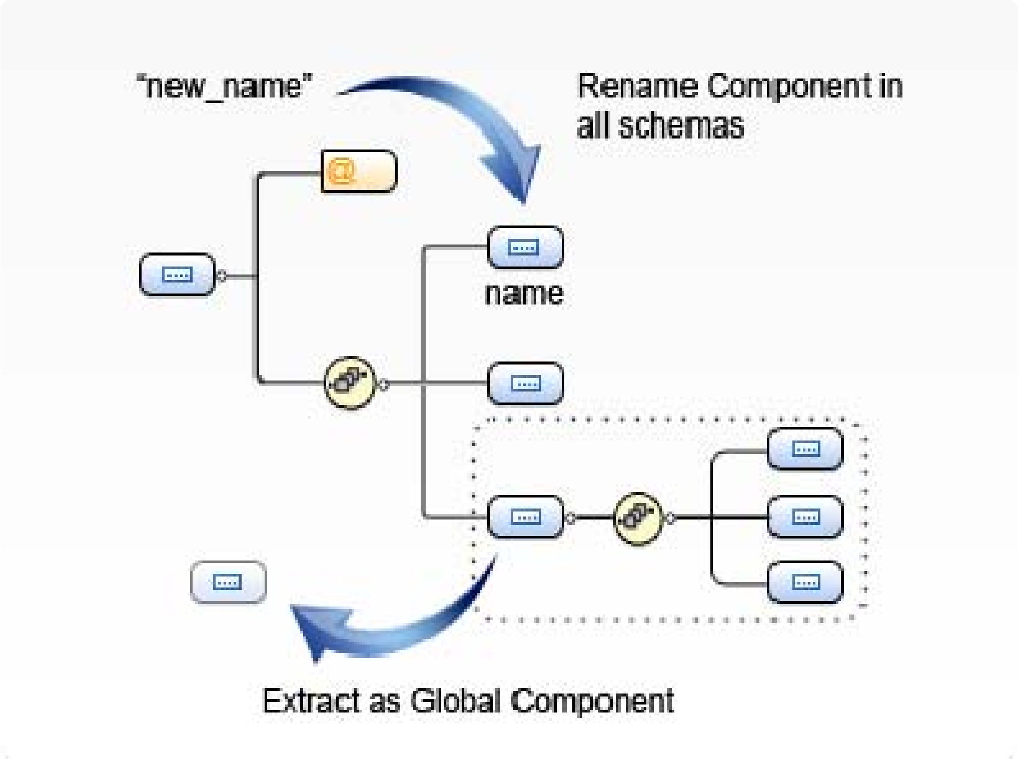
- Conventional wisdom in software engineering is to design for change as this reduces costs later in the life cycle.
- XP, however, maintains that this is not worthwhile as changes cannot be reliably anticipated.
- Rather, it proposes constant code improvement (refactoring) to make changes easier when they have to be implemented.

Refactoring

Refactoring

What is Refactoring?

- Changing the *structure* of the code without changing its *behavior* for better understanding
  - Example refactoring
    - Rename
    - Extract method/interface
    - Inline
    - Pull up/Push down



Ref: http://www.oxygenxml.com/img/mb_dev_schema_refactoring.png

Key points

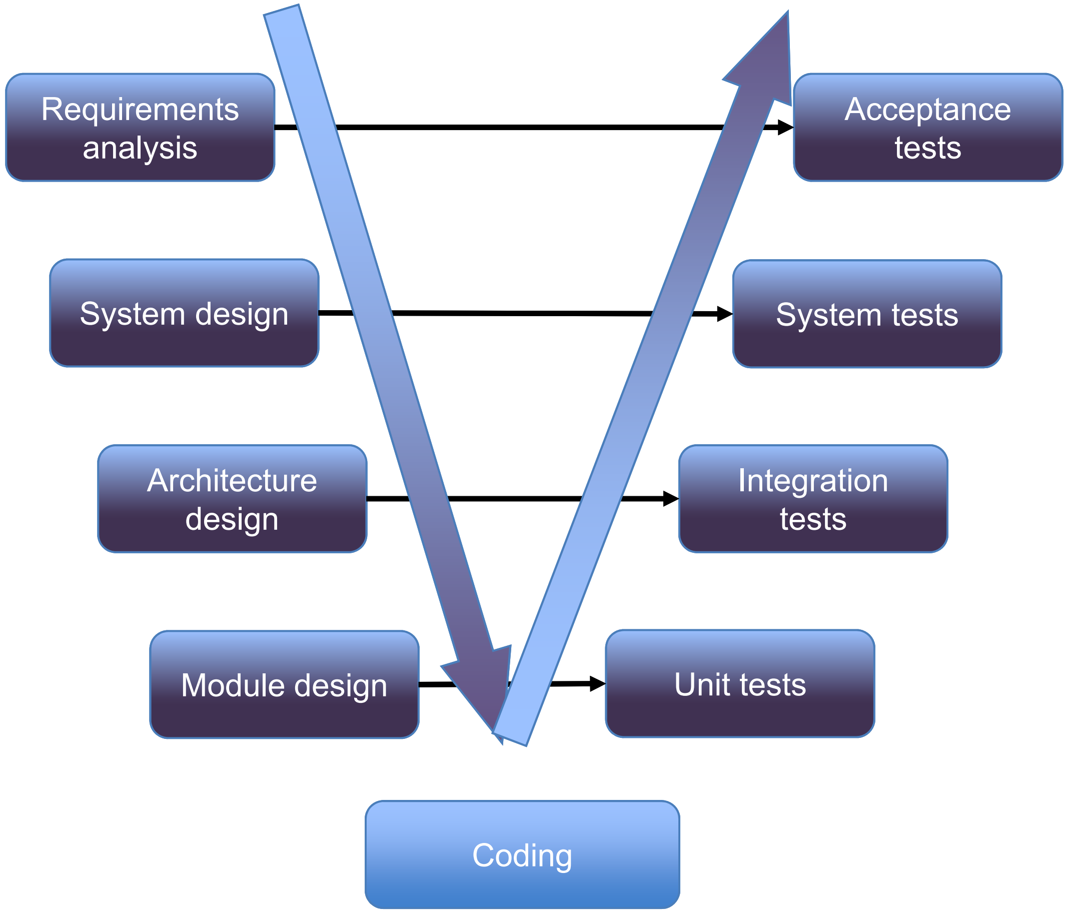
- Agile methods are incremental development methods that focus on:
  - Rapid development
  - Frequent releases of the software
  - Reducing process overheads and producing high-quality code.
- Use an agile or a plan-driven approach depends on:
  - The type of software being developed
  - The capabilities of the development team and the culture of the company developing the system.
- Extreme programming:
  - A well-known agile method that integrates a range of good programming practices

Chapter 3 – Agile Software Development

Lecture 2

Testing in XP

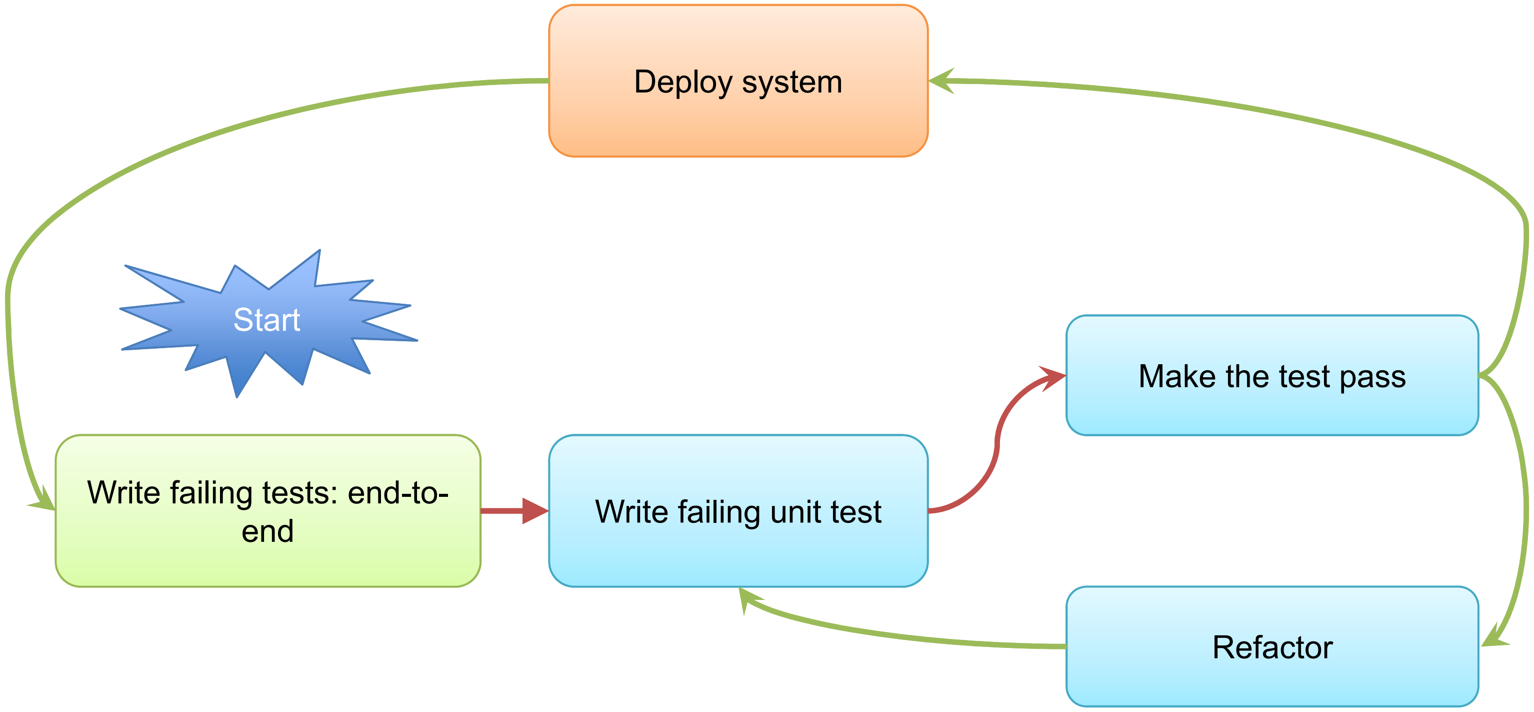
- Testing features:
  - Test-first development
  - Incremental test development
  - User involvement in test validation
  - Automated test harness to run all tests each time a new release is built



Ref: http://fileadmin.cs.lth.se/cs/Education/EDA270/Reports/2009/SvenssonGraden.pdf

Test-first development

- Writing tests before code clarifies the requirements to be implemented.



Customer involvement in XP

- The role of the customer in the testing process is to help develop acceptance tests for the stories that are to be implemented in the next release of the system.
- All new code is therefore validated to ensure that it is what the customer needs.
- They may feel that providing the requirements was enough of a contribution and so may be reluctant to get involved in the testing process.

Test case description for dose checking

Test 4: Dose Checking

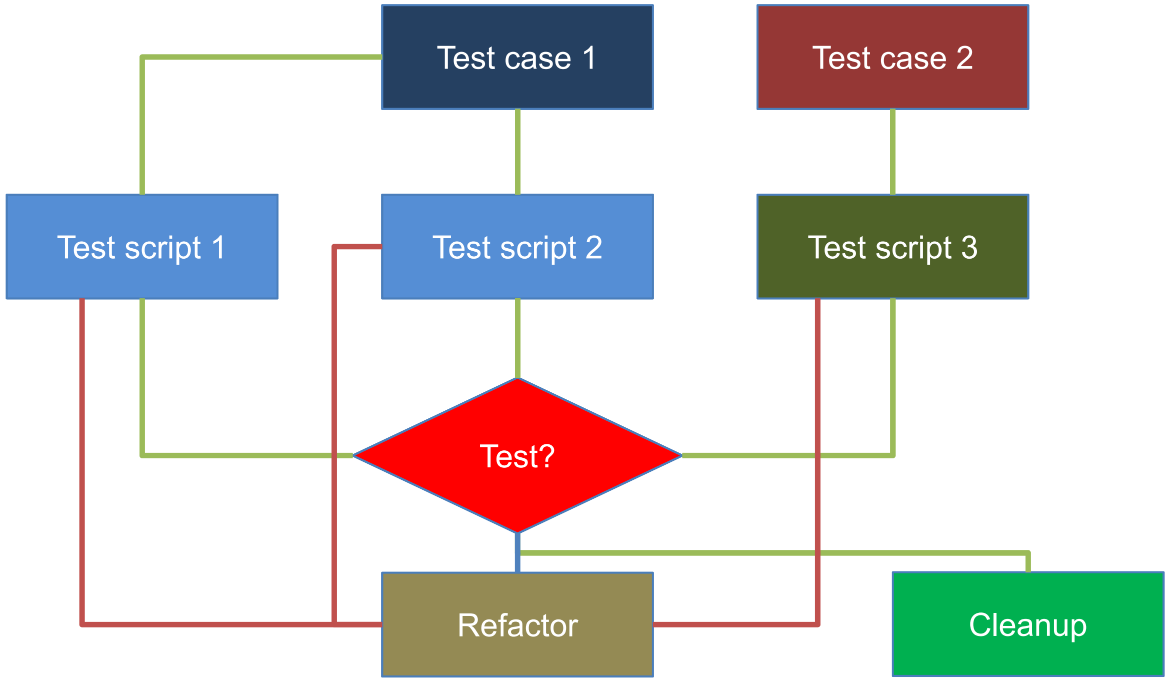
Input:
1: A number in mg representing a single dose of the drug.
2: A number representing the number of single doses per day.

Tests:
1: Test for inputs where the single dose is correct but the frequency is too high.
2: Test for inputs where the single dose is too high and too low.
3: Test for inputs where the single dose frequency is too high and too low
4: Test for inputs where the single dose frequency is in the permitted range.

Output:
Ok or error message indicating that the dose is outside safe range

Test automation

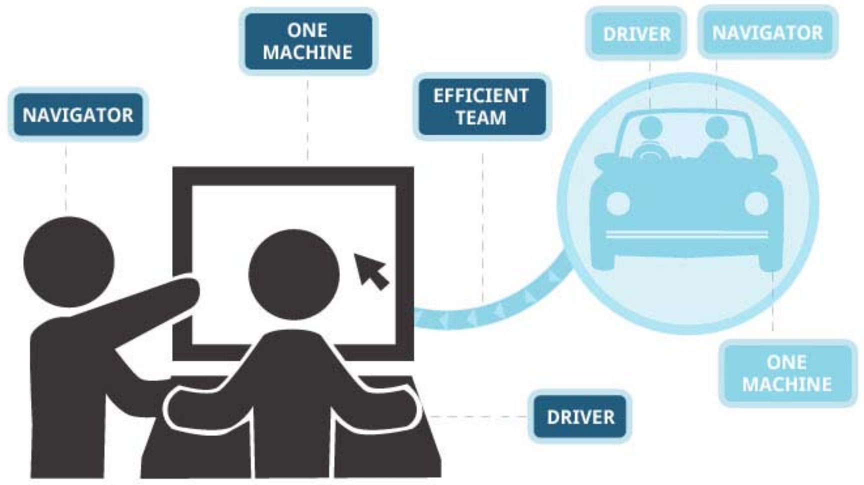
- Test automation means that tests are written as executable components before the task is implemented



XP testing difficulties

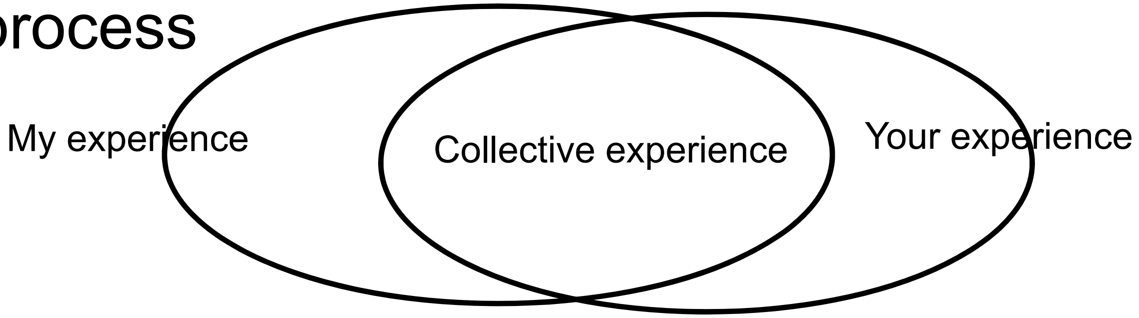
- Programmers prefer programming
- Taking short cuts when writing tests
- Writing incomplete tests that do not check for all possible exceptions that may occur.
- Difficult to write test that work incrementally
  - For example unit tests for the code that implements the 'display logic' and workflow between screens.
- It is difficult to judge the completeness of a set of tests.

Pair Programming



Pair programming

- Two people working together at a single computer.
- 2 people working at a single computer will add as much functionality as two working separately except that it will be much higher in quality.
- A review process



Pair programming

- In pair programming, programmers sit together at the same workstation to develop the software.
- Pairs are created dynamically so that all team members work with each other during the development process.
- The sharing of knowledge that happens during pair programming is very important as it reduces the overall risks to a project when team members leave.
- Pair programming is not necessarily inefficient and there is evidence that a pair working together is more efficient than 2 programmers working separately.

Advantages of pair programming

Review

ALL OF THESE PARENTHESES AND SEMICOLONS TOTALLY RUIN THE FENG SHUI OF THE CODE.
SALLY LEARNS AN IMPORTANT LESSON ABOUT INVITING THE RIGHT PEOPLE TO HER CODE REVIEWS.

Idea sharing

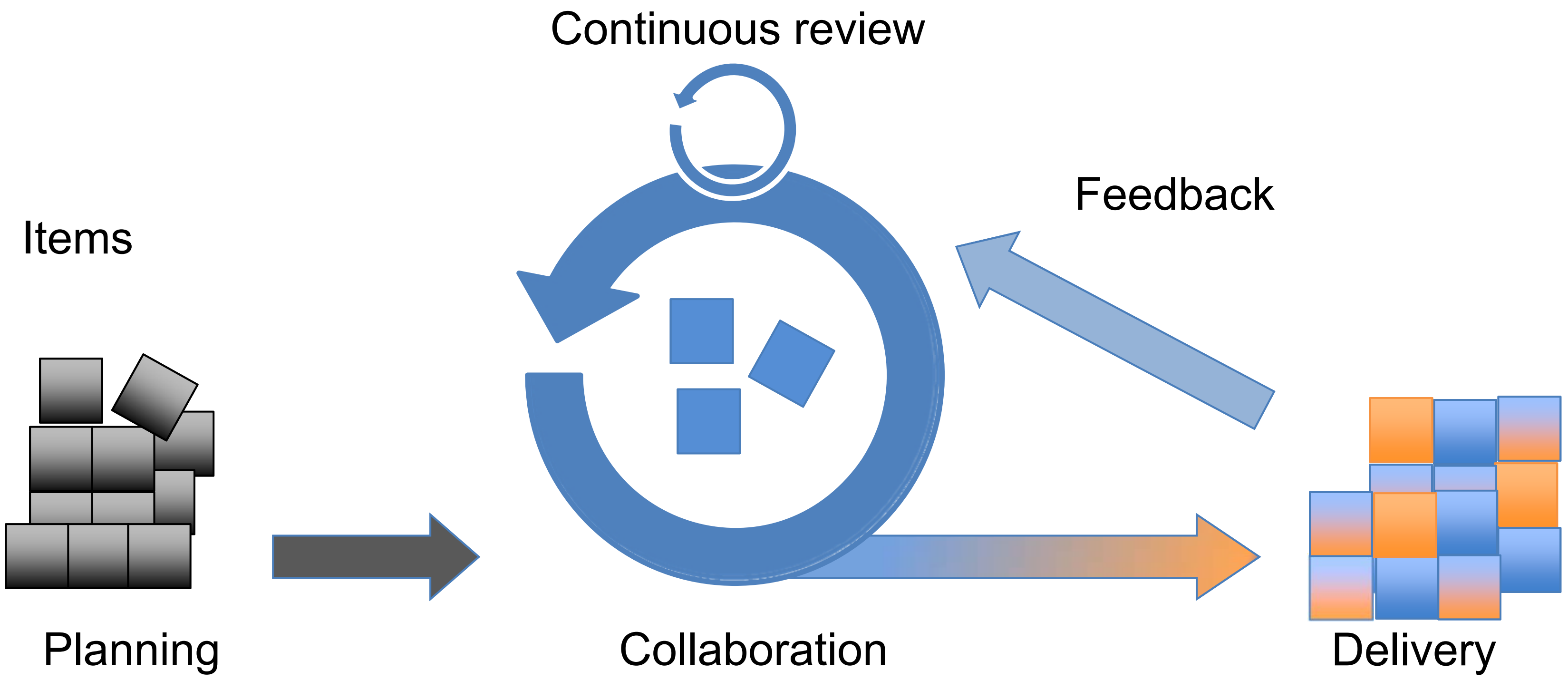
Dynamicity

Responsibility

Scrum

Agile project management (SCRUM)

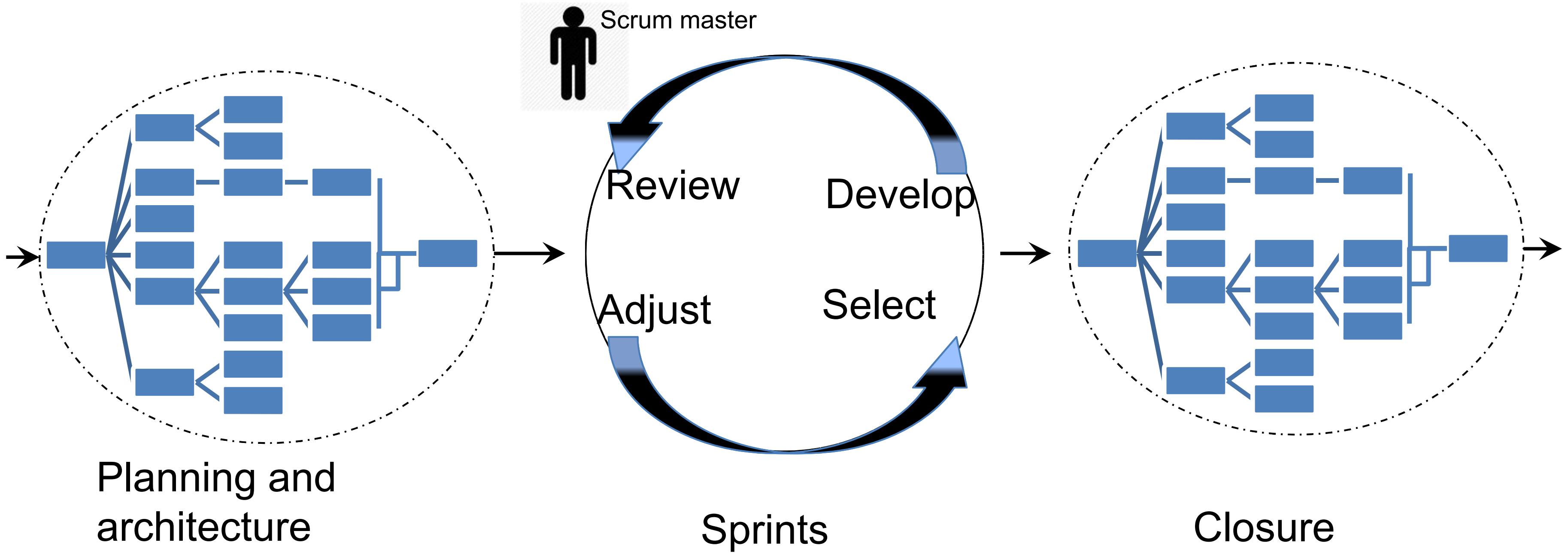
- The Scrum approach is a general agile method but its focus is on managing iterative development rather than specific agile practices



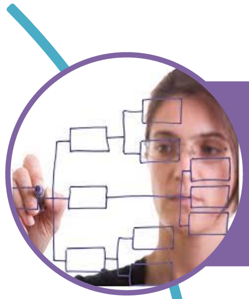
Ref: http://en.wikipedia.org/wiki/Project_management

The Scrum process

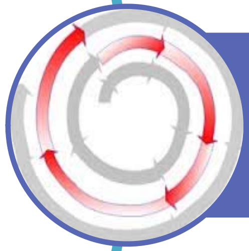
- The Scrum approach is a general agile method but its focus is on managing iterative development rather than specific agile practices.
- Three phases



The Scrum Process: Explained



Planning and architecture: Establishing objectives and design architecture



Sprint cycles: Each sprint develops an increment of system

Closure: Wrap up of project, documentation, such as manuals and lessons learned completion

Sprint cycle

Length
- Normally 2-4 weeks
- Corresponds to a release development of XP

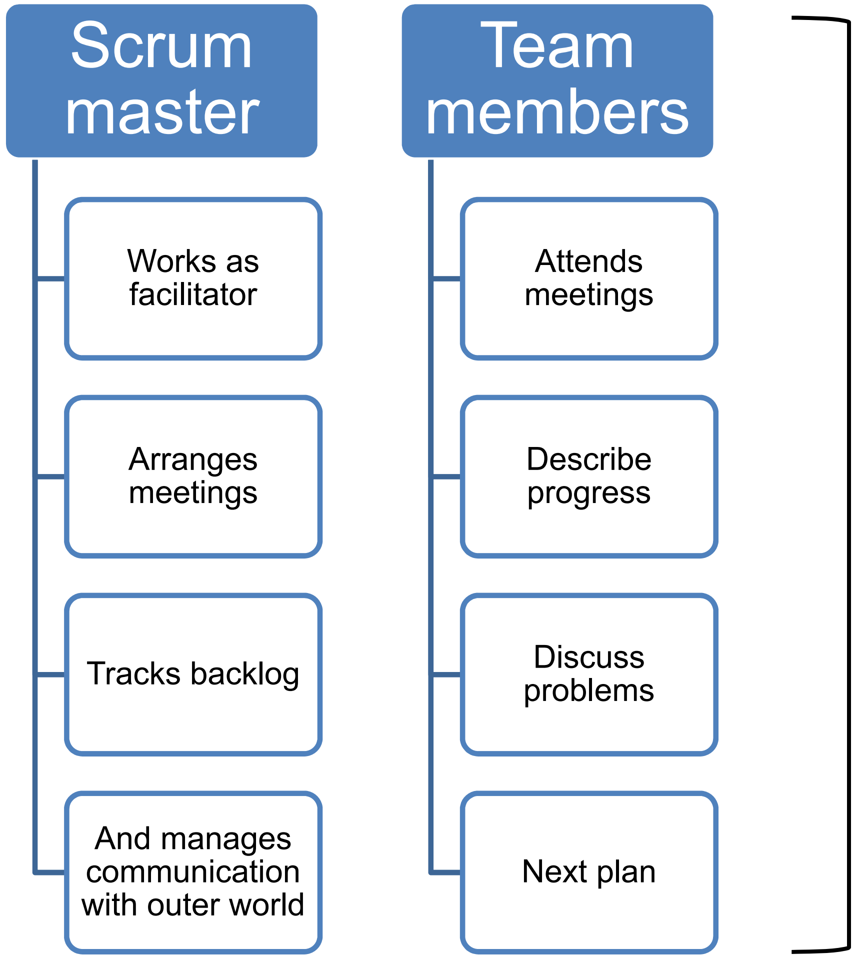
Adjust
- Start point: backlog of product
- Backlog is a list of work to be done

Select
- Involves project team to select features etc.
- Develop functionality

Review
- Reviewed and shown to stakeholders
- Next cycle begins

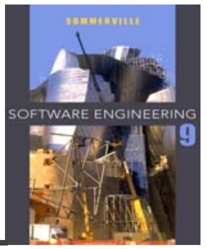
Master
- Manage all communication
- Protect from external distraction

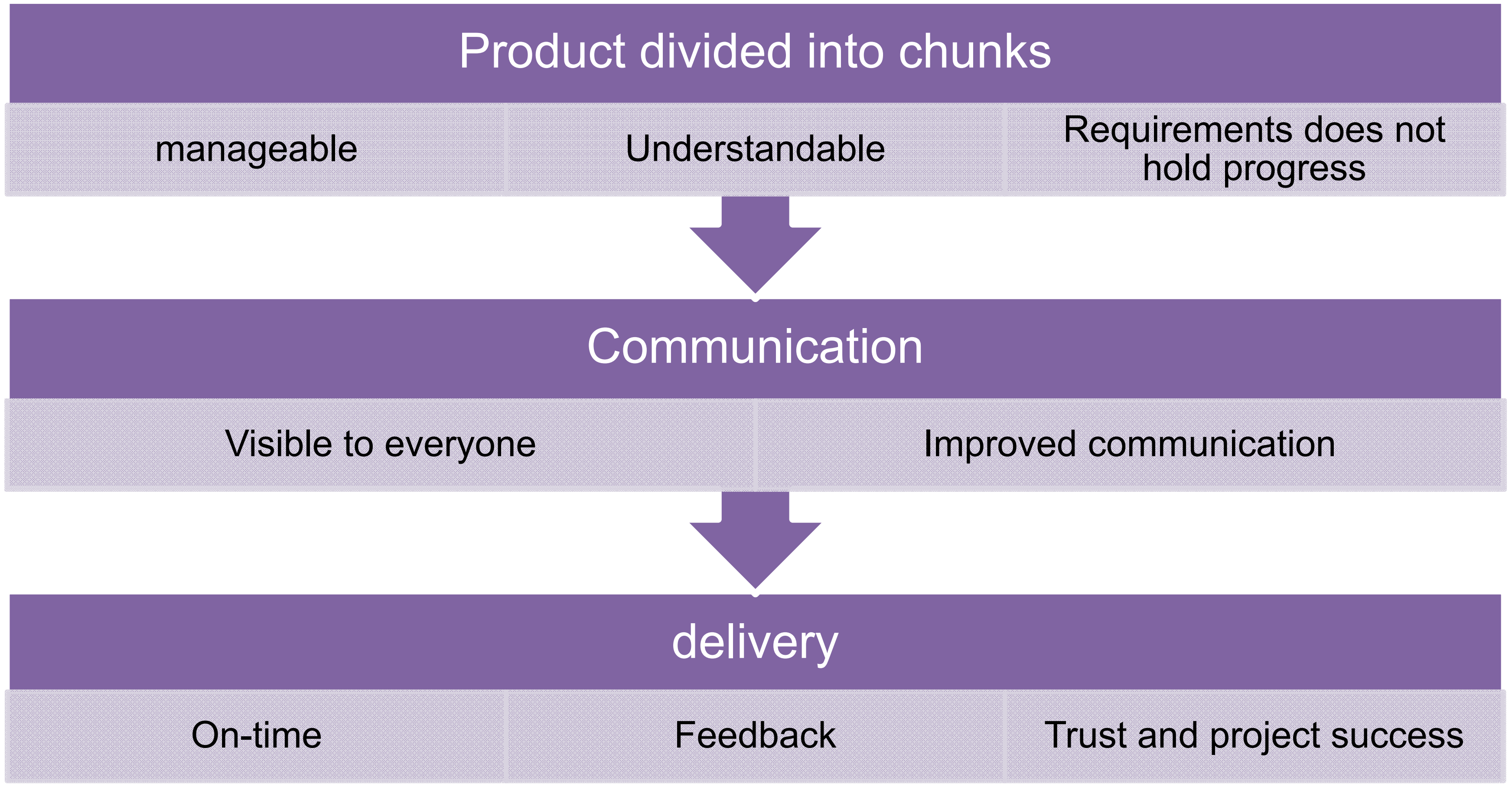
Teamwork in Scrum



Team work

Scrum benefits



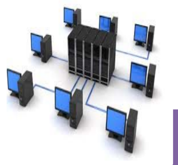


Scaling Large/Long Systems

Scaling agile methods

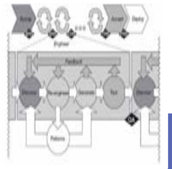
- Agile methods have proved to be successful for small and medium sized projects that can be developed by a small co-located team.
- It is sometimes argued that the success of these methods comes because of improved communications which is possible when everyone is working together.
- Scaling up agile methods involves changing these to cope with larger, longer projects where there are multiple development teams, perhaps working in different locations.

Large systems development



Collection of separate systems

- Independent developed
- In different places
- In different zones



Brownfield systems

- Interaction with other systems
- Systems interaction

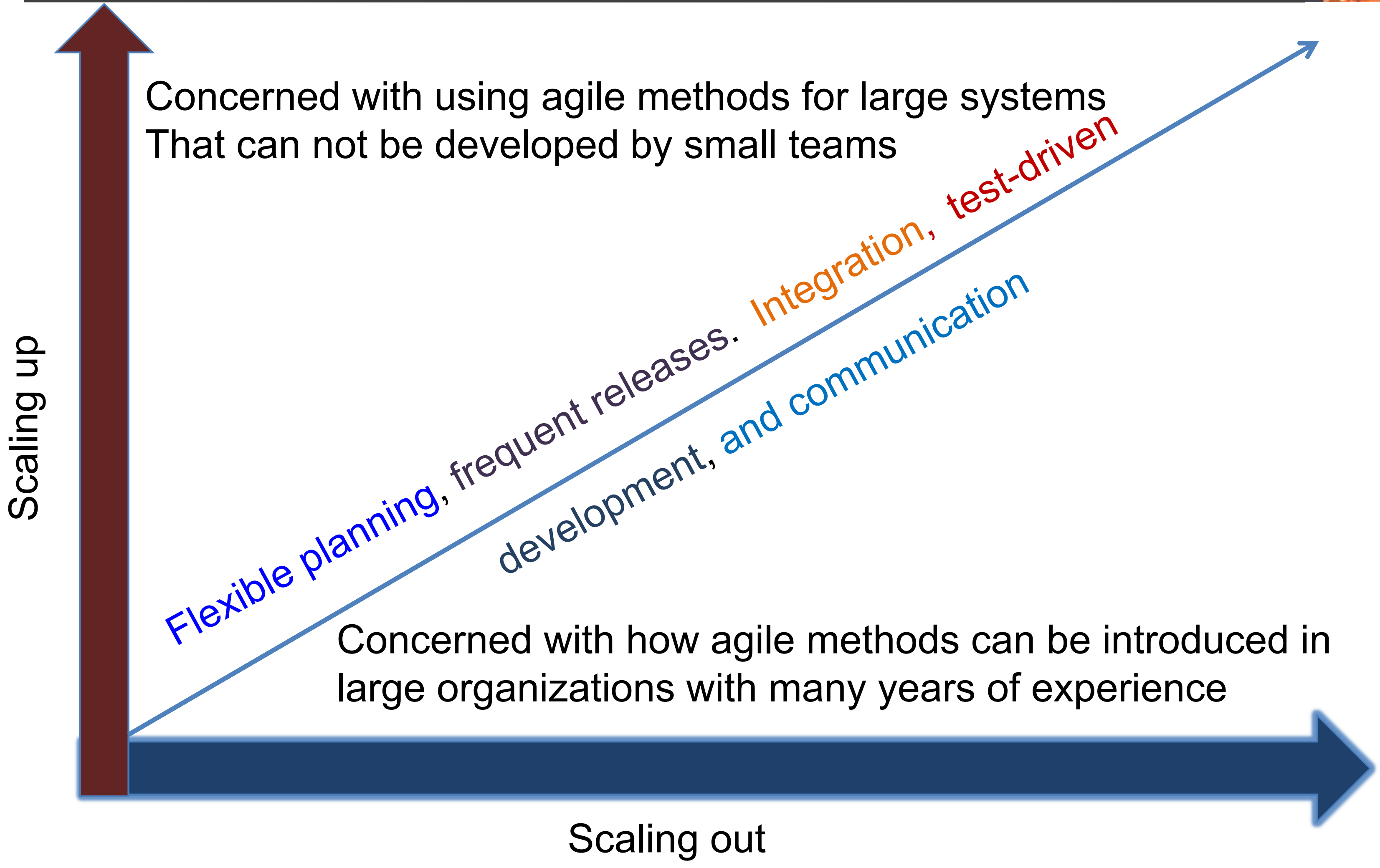
Concerned with configuration

- Part of system concerned with configuration
- Along with code development

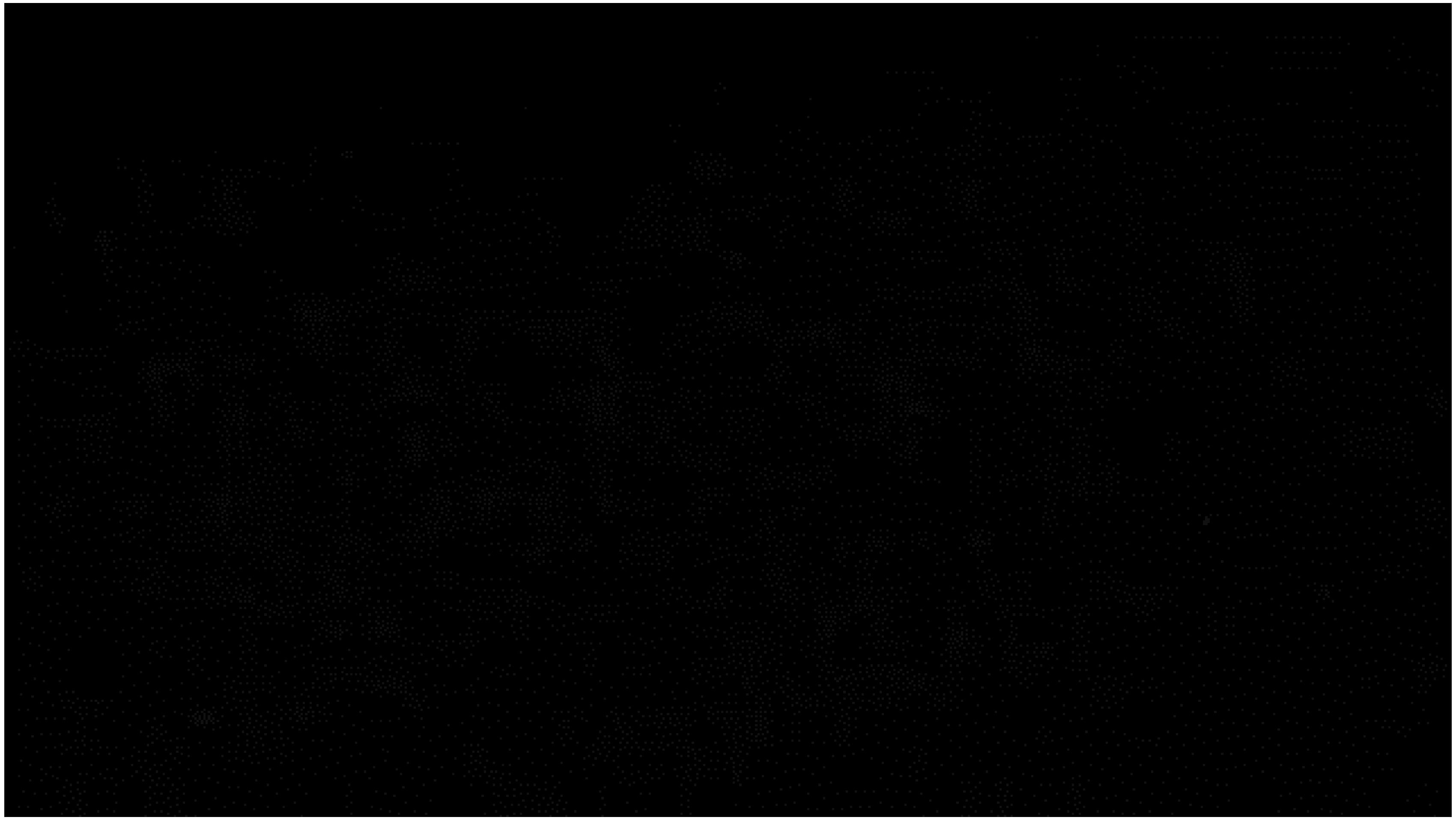
Large systems development

- Constraints:
  - Often constrained by external rules and regulations and limiting the way they can be developed
- Time:
  - long development time, difficult to maintain coherent teams. people move to other job etc.
- Stakeholders:
  - Many stakeholders, difficult to involve all in the development process

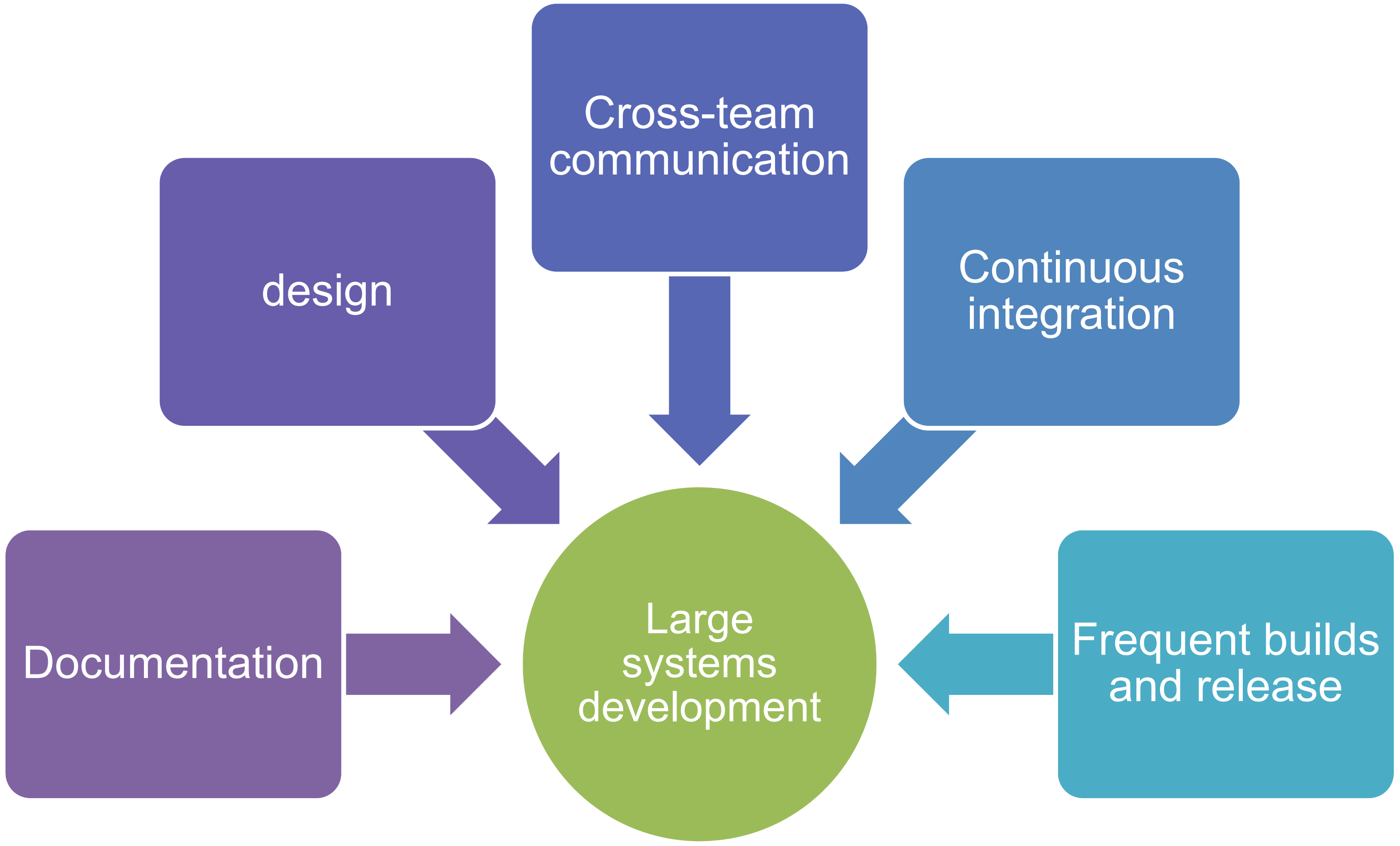
Scaling out and scaling up


Concerned with using agile methods for large systems
That can not be developed by small teams
Flexible planning, frequent releases. Integration, test-driven
development, and communication
Concerned with how agile methods can be introduced in
large organizations with many years of experience
Scaling up
Scaling out

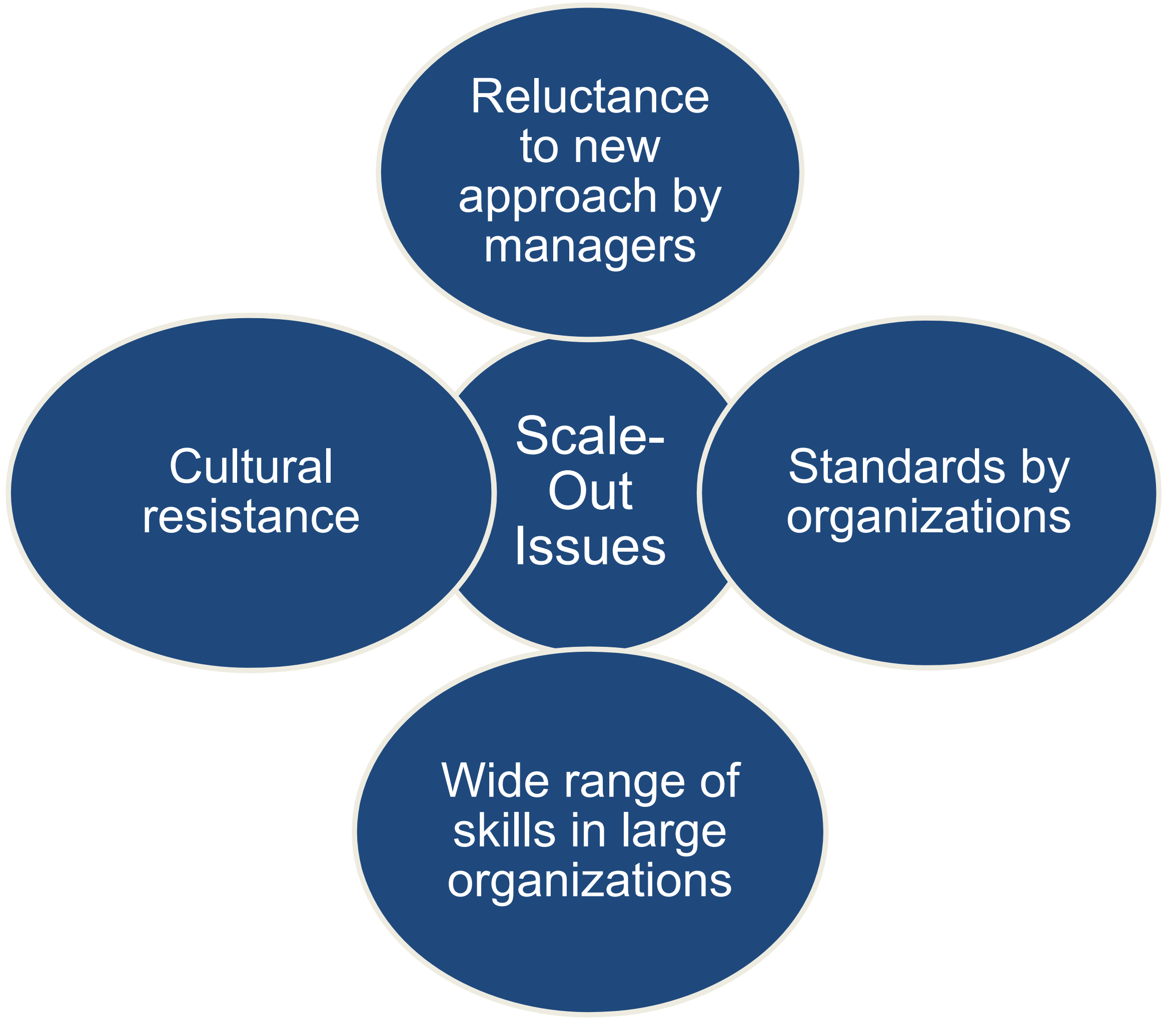
Scaling out Or scaling up?



Scaling up to large systems: Requirements



Scaling out to large companies: Issues


Reluctance to new approach by managers
Cultural resistance
Scale-Out Issues
Standards by organizations
Wide range of skills in large organizations

Scaling out to large companies

- Project managers who do not have experience of agile methods may be reluctant to accept the risk of a new approach.
- Large organizations often have quality procedures and standards that all projects are expected to follow and, because of their bureaucratic nature, these are likely to be incompatible with agile methods.
- Agile methods seem to work best when team members have a relatively high skill level. However, within large organizations, there are likely to be a wide range of skills and abilities.
- There may be cultural resistance to agile methods, especially in those organizations that have a long history of using conventional systems engineering processes.

Key points

- A particular strength of extreme programming is the development of automated tests before a program feature is created. All tests must successfully execute when an increment is integrated into a system.
- The Scrum method is an agile method that provides a project management framework. It is centred round a set of sprints, which are fixed time periods when a system increment is developed.
- Scaling agile methods for large systems is difficult. Large systems need up-front design and some documentation.