Diploma in

Computer Science

Agile Software Development Methods

Objectives

- Appreciate the shortcomings of traditional software design methodologies
- Examine how agile software development methodologies speed up software development
- Understand how different software development strategies under agile development are used
- Distinguish between different software development strategies

Agile development methods

Life before agile

Life before agile

- Faster than they had ever moved in the entire history of computing.
- Because they simply didn't meet the needs of the business anymore.
- Projects could stretch on for as long as 20 years.

Life before agile

- The information used on the 1982 space shuttle was from the 1960s!
- In today's terms, it would be unthinkable to develop any kind of software for that long.

Life before agile

- Other industries also started implementing agile development methods.
- In the 1990s, it took the automotive industry six or more years to design a new car from scratch.

The agile era

The agile era

- Frustrations lead to the Utah meeting in early 2001
- Rapid feedback, customer engagement, and incorporating changes became key
- Very little is set in stone at the beginning of the project
- Focus is to get to work as soon as possible

Old vs new

- Linear methods of old seek to set the budget, requirements, and scope at the very start of the project and in as much detail as possible.
- If for some reason something in the project has to change, it means that all that research has to be done again.
- Linear methods are not really built with changing requirements in mind

Old vs new

- Agile development incorporates changes at every iteration of the product.
- There isn't really a final product until the customer is satisfied.
- There is something that is delivered to the customer at every milestone.
- Linear methods put emphasis on the planning rather than on delivering actual work.

Agile programming principles

- Particularly known for lower cost
- Constant feedback from the customer
- Less likely to get the solution wrong since client is extensively involved

Quick and easy adaptation

- Extremely easy for teams to adapt to change
- Customers typically change their requirements and team members should be able to adapt to these

Technical debt

- Maintenance tasks that are required to support a product
- Any such documentation is added to the product backlog and attended to in each sprint
- Used to determine what needs to be addressed next

Minimising risk

- Agile methods have functioning code at a very early stage
- Relatively easy to go back and fix problems

Higher quality

- Features are not developed at the same time
- Subsets of the features are developed in sprints
- Developers have more time to perfect them before a release

Predictable release dates

- Built around shorter working times
- Production of a tangible product at the end of each working period
- Easier to predict delivery dates

Collaboration and communication

- Built around people working in teams
- There's a far less chance of unplanned changes to the project
- More control of the direction of the project and more focus on what the client wants

Key principles of the agile era

Principle	Description
Satisfaction and delivery	Customer satisfaction through early and continuous working software
Welcoming change	Welcome changing requirements, even at later stages of development
Deliver frequently	Deliver working software frequently (weekly rather than monthly)
Communication is key	Ensure close association of developers with businesspeople on a daily basis
Environment and trust	Build projects around motivated individuals, give them support and trust them
Face-to-face communication	Encourage face-to-face conversation to ensure efficient and effective communication
Software as measure of progress	Working software is the primary measure of progress
Sustainable development	Maintain a constant pace throughout the development
Attention to detail	Continuous attention to technical excellence and good design
The power of less	Simplicity is essential
Self-organising teams	Team’s attention is on being effective in changing circumstances

Agile frameworks

Idea built upon iterative development

At the end of each stage, there is a tangible delivery

Work is started as soon as possible

Product is improved by looking at it repeatedly

Scrum

- Lightweight process framework which uses the agile methodology
- Requires the overheads of the processes to be kept as small as possible
- Focus is a working software package
- Has distinct steps

Scrum

Three categories:

- Roles
- Artefacts
- Time boxes

Scrum roles table

Product owner	Scrum Master	The team
Defines features of the product	Manages the team and looks after the team's productivity	Have approximately 5-9 members
Decides the release date and corresponding features	Maintains the block list and removes barriers in development	Includes developers, designers and sometimes testers, etc.
Prioritises the features according to the market value and profitability of the product	Coordinates with all roles and functions	Organises and schedules their work on their own
Responsible for the profitability of the product	Shields team from external interferences	Can do everything within the boundaries of the project to meet the sprint goal
Can accept or reject work item result	Sends invites to the daily scrum, sprint review and planning meetings	Actively participate in daily ceremonies

Scrum artifacts

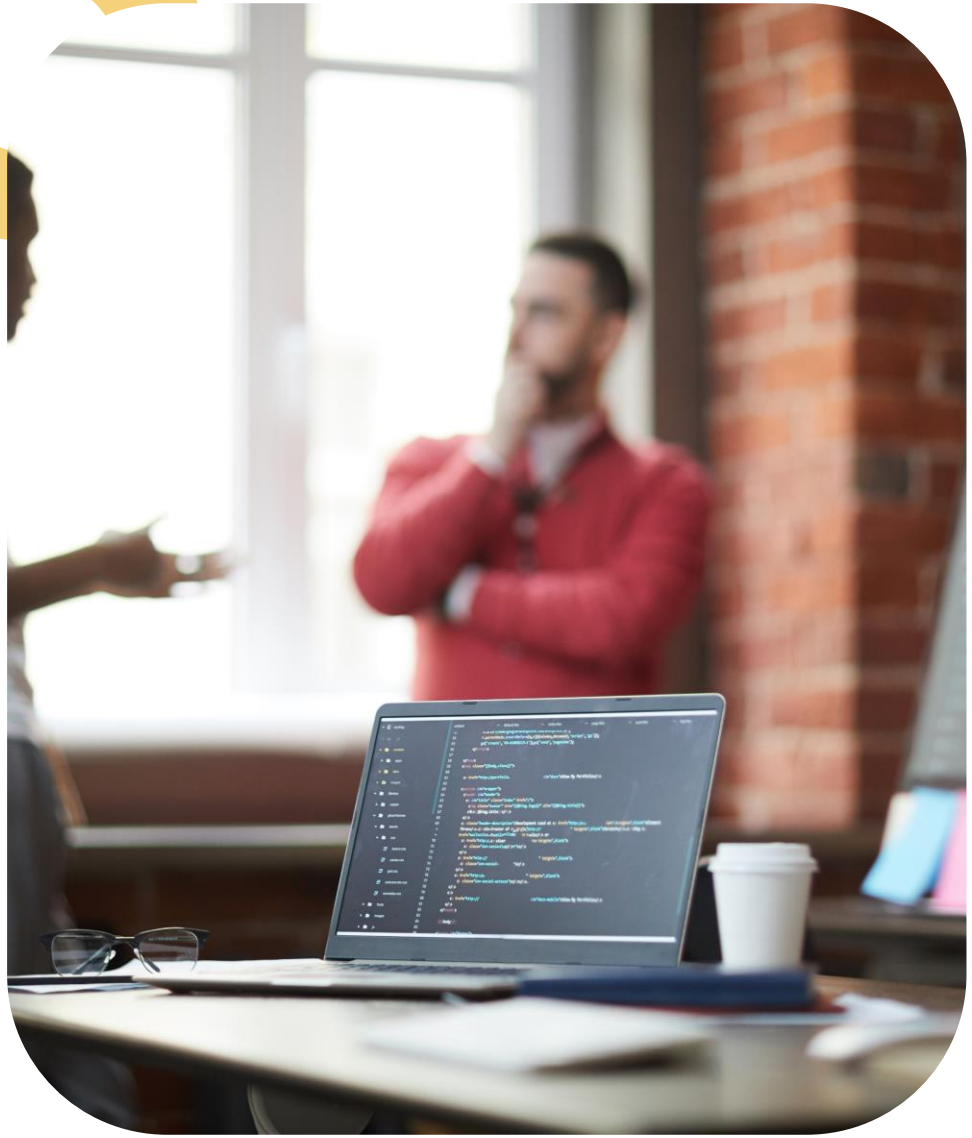
- Represent the work
- Put everyone on the same page
- Constitutes a few activities
- User stories

Scrum activities

- Product backlog
- Release backlog
- Sprints
- Sprint backlogs
- Blocklists
- Burndown charts

Product backlog

A collection of user stories prepared and approved by the product owner.



Release backlog

A list of the stories which should be slotted for release.

Ideally all the stories in the backlog are attended to and completed by the time we get to the next release.

Sprints

- Actual work in the release backlogs is done
- Typically decided by the product owner
- Completion of user stories in the product backlog

Sprint backlog

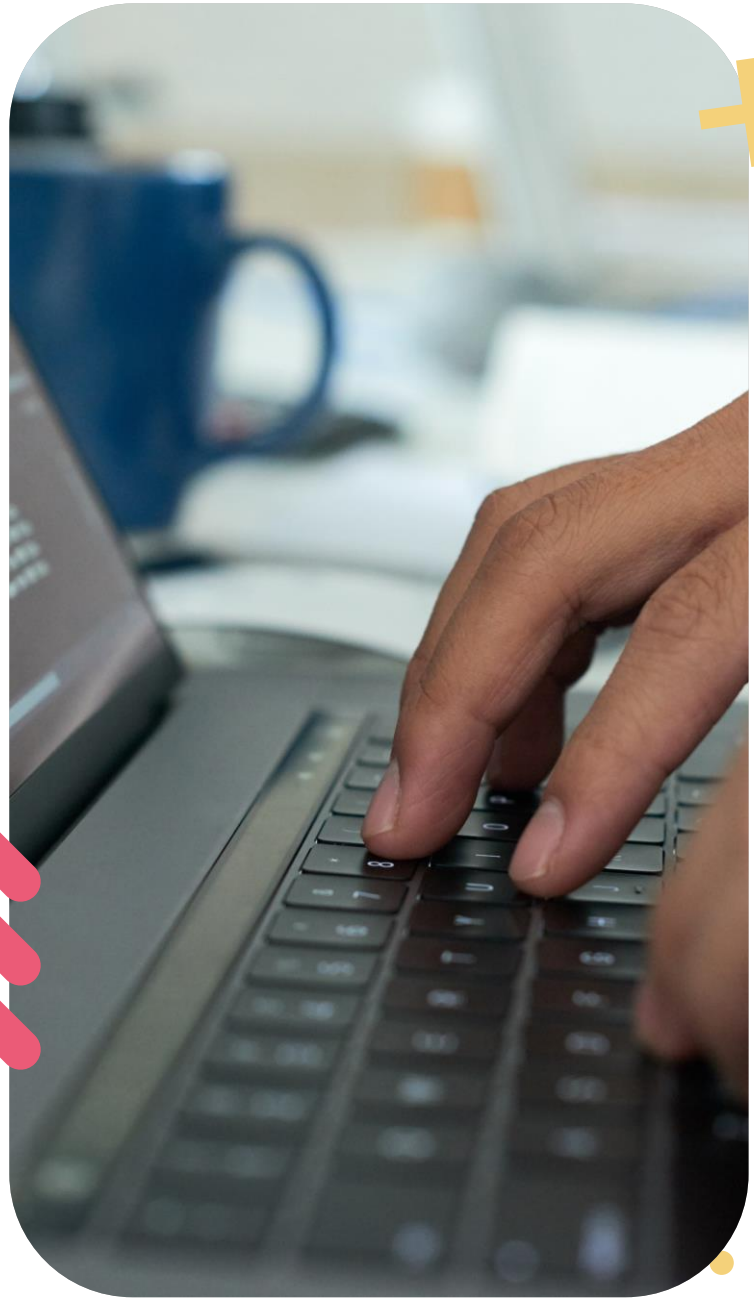
- This is the list of user stories that are set to be completed in a sprint
- Updated daily
- Team members take up work on their own

Burndown chart

- The overall progress of work in the team
- Reflection of the overall process
- Typically presented in a graph

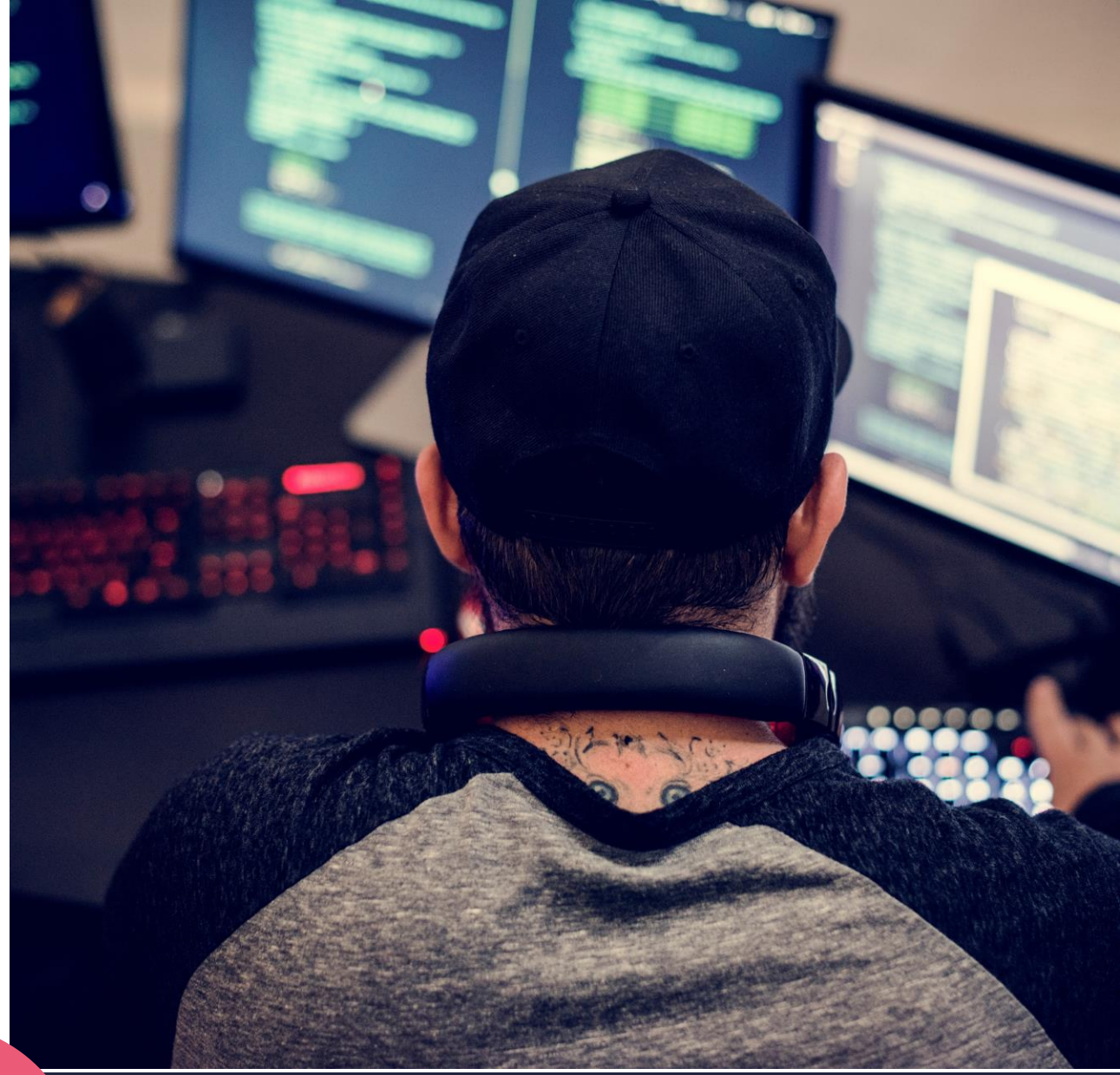
Ceremonies

- Processes that result in milestones in the scrum process
- Sprint planning
- Meetings hosted by the scrum master



Daily scrum

- Short meeting (max 15 mins)
- Hosted by the scrum master
- This is used as a tracking method for the team’s progress



Sprint review

A 2-4 hour meeting hosted by the scrum master to discuss the accomplishments of the team and lessons learned in the past sprint.

Scrum master

Sort of an overseer on the project

Ensures that all the team members follow the scrum’s theories, rules, practices, and activities

Makes sure that any obstacles are removed

Facilitates scrum meetings and communicates with the product owner

Product owner

- Manages product backlog
- Collaboration with development team and stakeholders

Developers

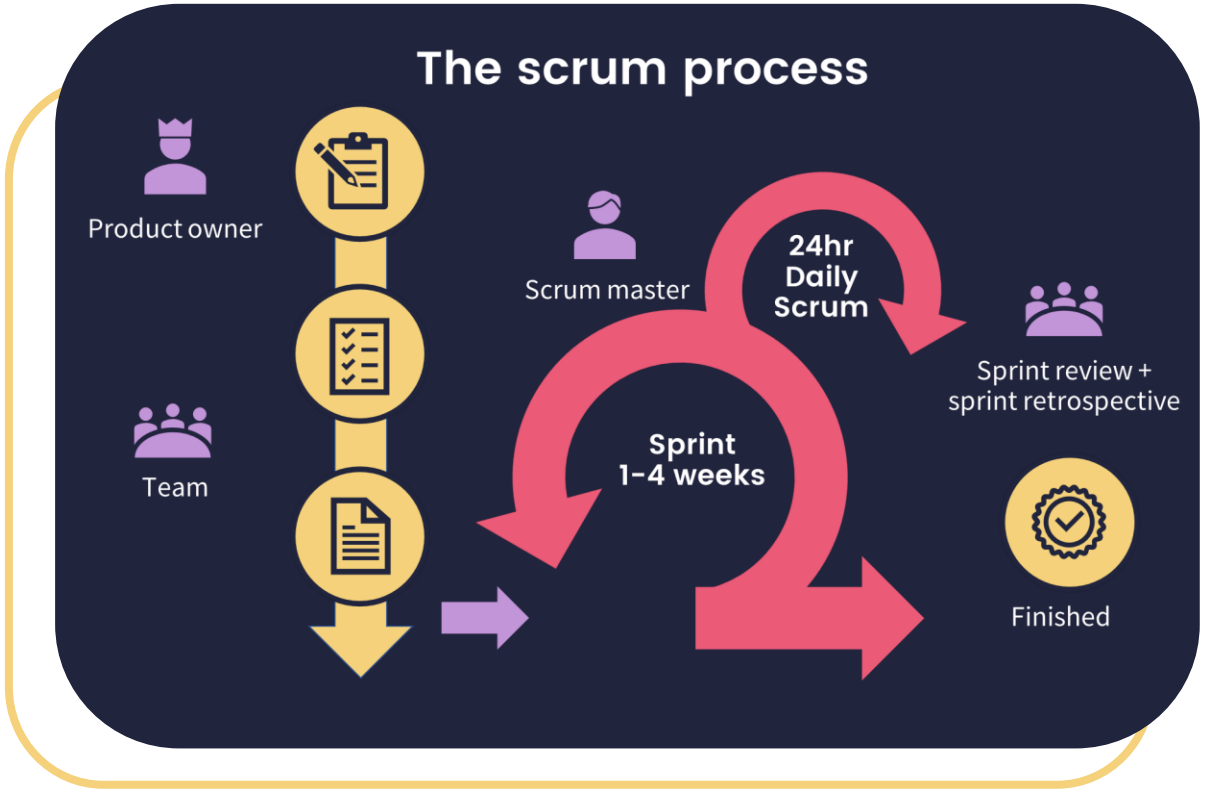
- Comprised of developers proficient in their own area of expertise
- Work on the actual implementation of the deliverable software which is to be delivered at the end of each Sprint

Development team

Two main responsibilities

- Development and delivery
- Tasking and providing estimations

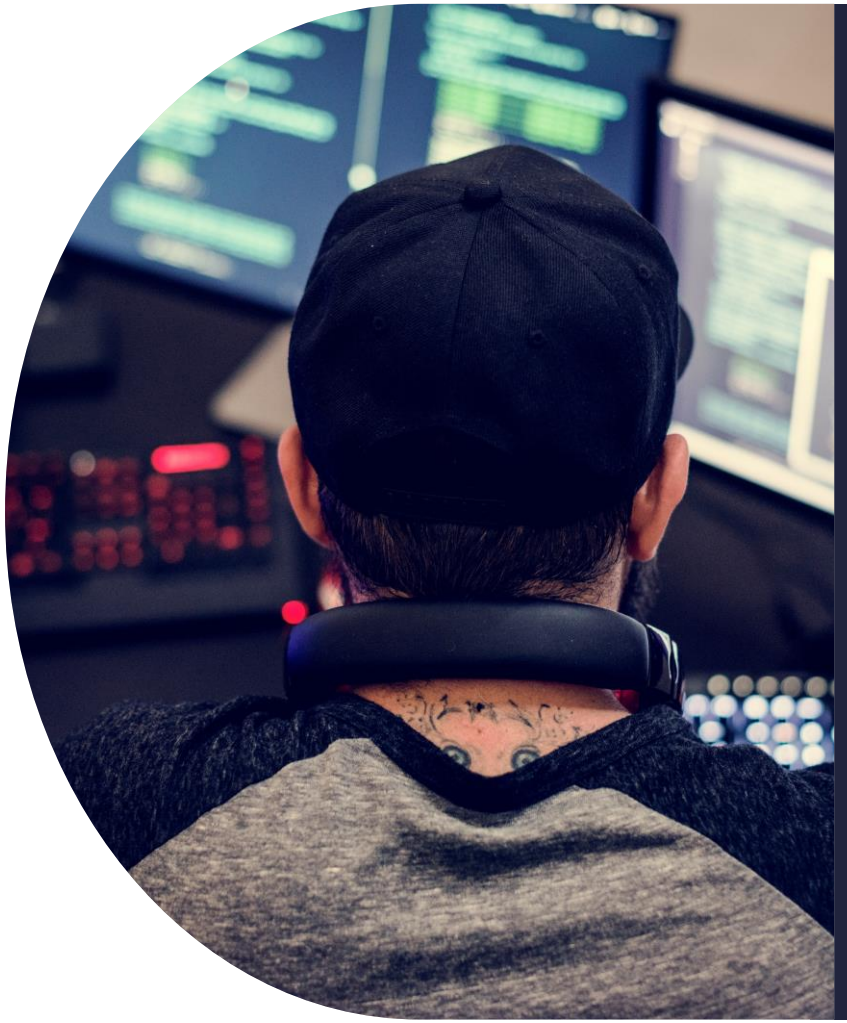
The scrum process

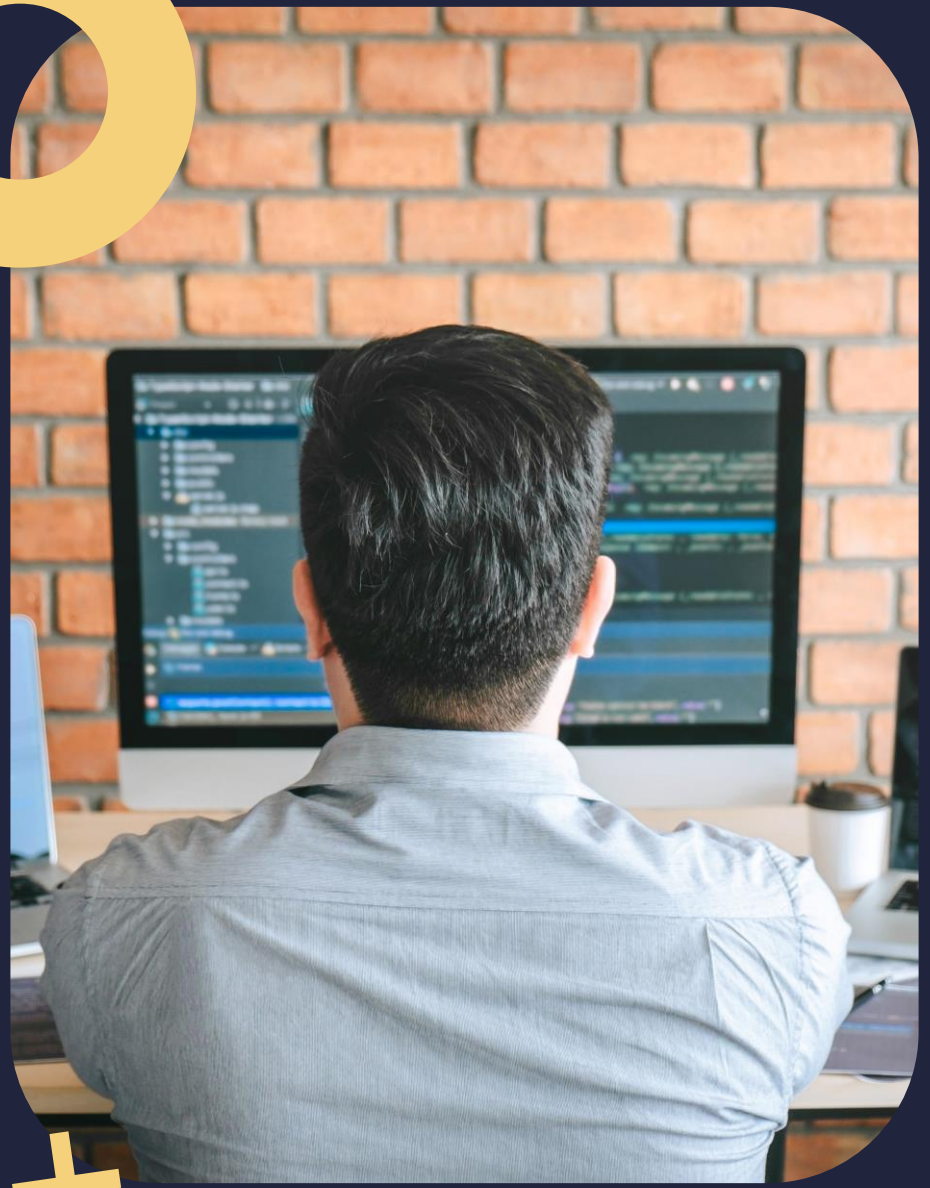


- Product owner-list of tasks
- Scrum planning-first product backlog-sprint backlog
- Sprint-sprint review meeting
- The deadline has been reached
- The budget is exhausted
- The product owner is satisfied with the final product

Extreme programming

- Lightweight
- Efficient
- Low-risk
- Flexible
- Predictable





Extreme programming

- Emphasis on continuous feedback from the customer
- Short iterations
- Design and re-design
- Frequent testing and coding
- Early elimination of defects to reduce costs
- Active customer involvement
- Delivering a working product to the customer

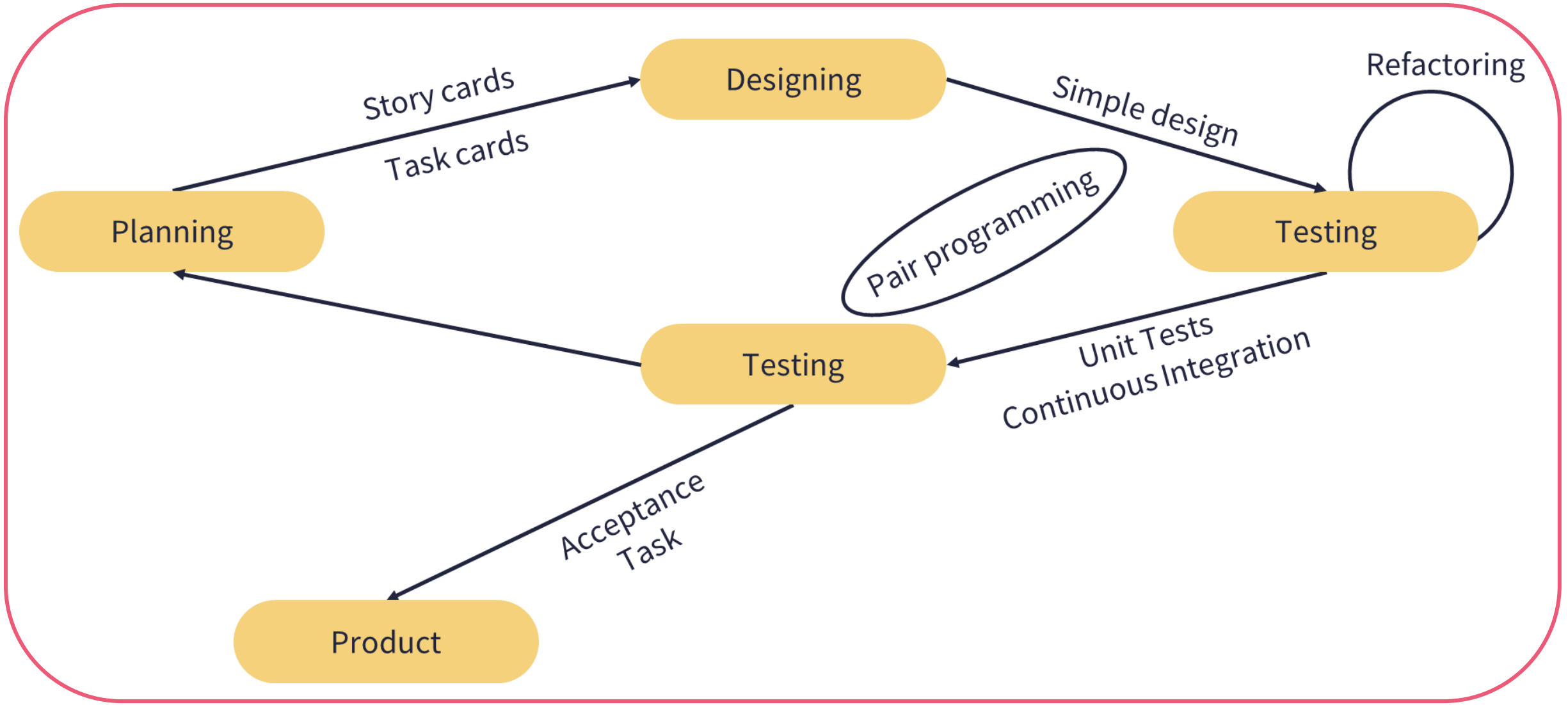
Extreme programming

- Literally takes effective principles and practises to extreme levels
- Code is reviewed all the time
- Continuous testing and regression
- Integration testing several times a day
- Alterations are short

Extreme programming

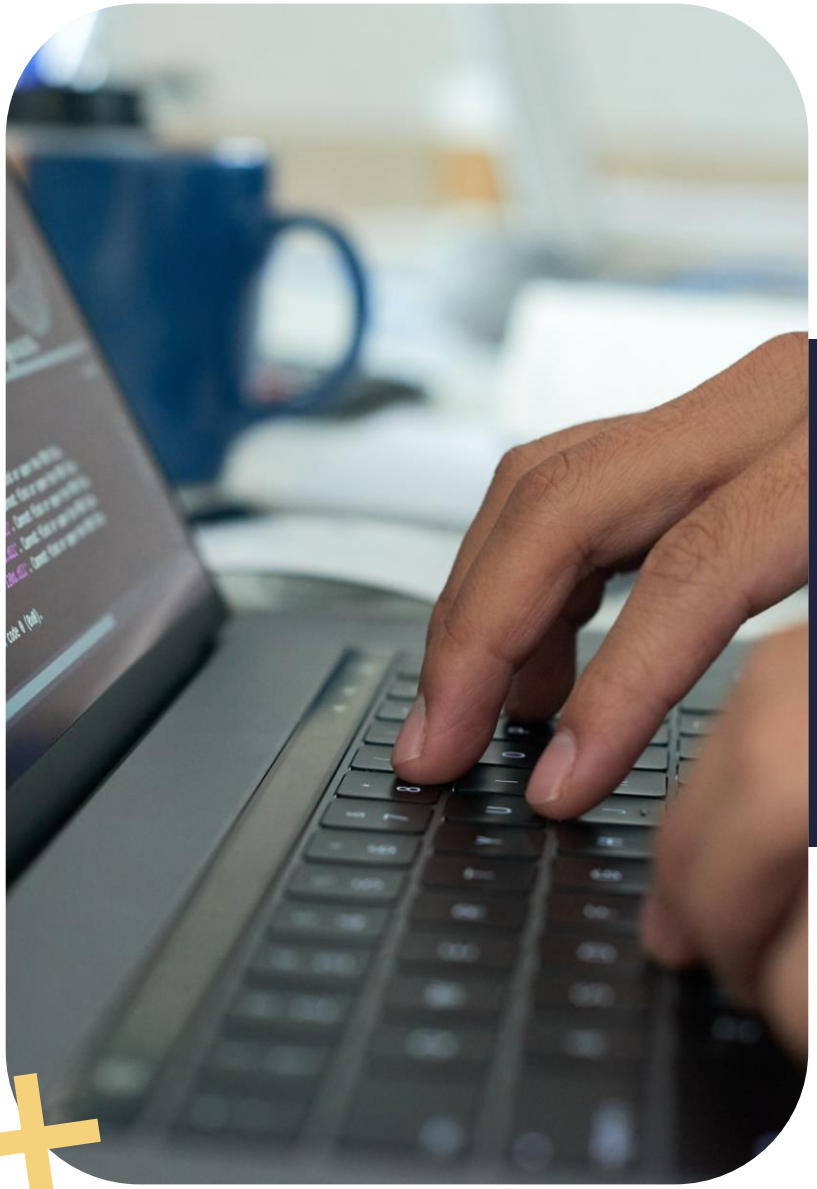
- It has slipped schedules and achievable development cycles
- Testing is extensive and ongoing
- Any existing functionality will not be broken by new features
- Communication is as effective as can be
- Reduces the overall project cost and boosts team cohesion and employee satisfaction

Extreme programming process



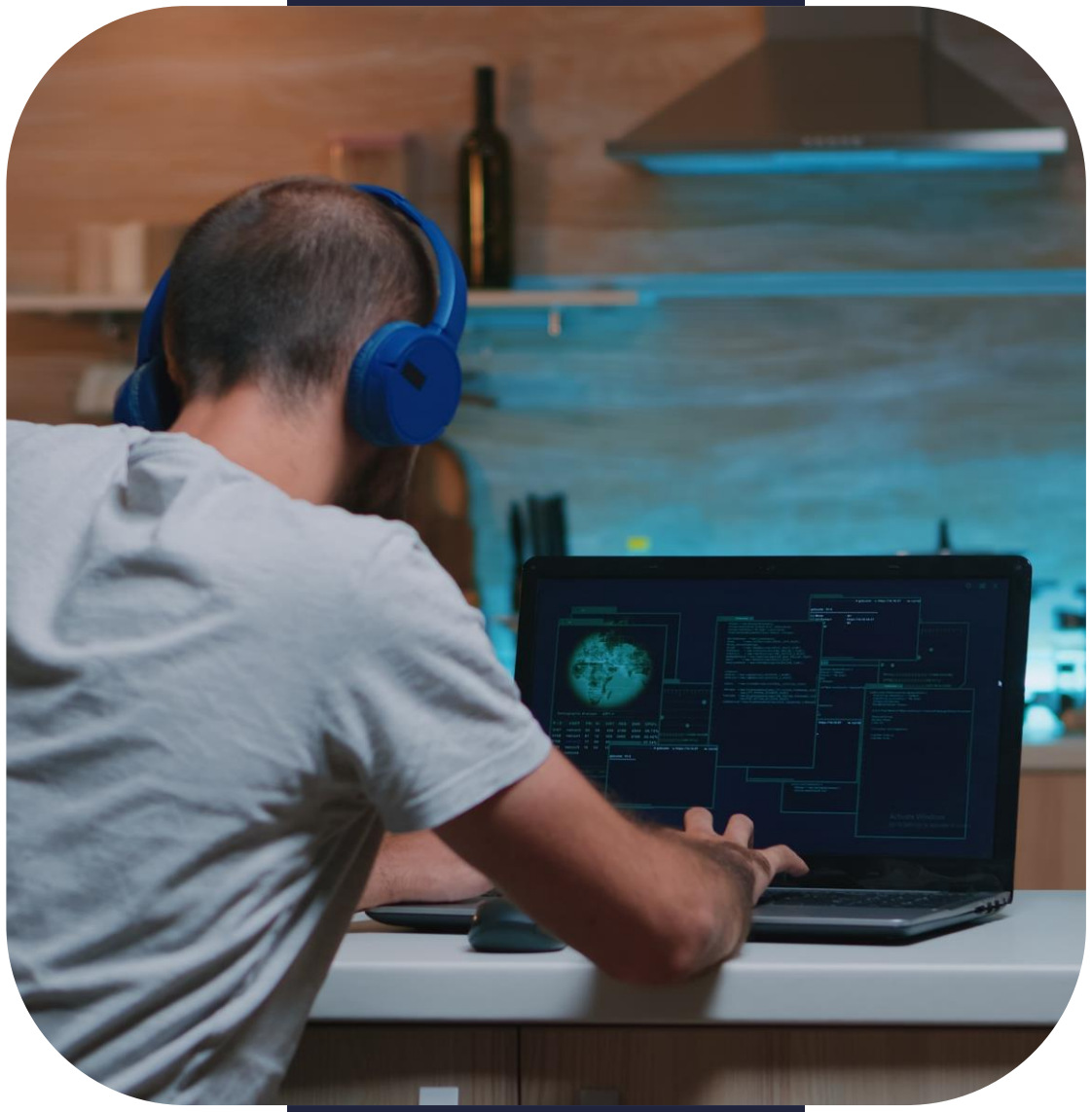
Extreme programming

- Much faster pace and mostly concurrent
- Produces software faster than any methods that we have looked at



Kanban

- Optimises the flow between the supplier and consumer
- Supermarket can still ensure that the product a consumer needs is always in stock



Kanban

- Transparency
- Balance
- Collaboration
- Customer focus
- Flow
- Leadership
- Understanding
- Agreement
- Respect

Kanban

Visualise

Commitment point

Delivery point

WIP limits

Manage flow

Kanban policies

- Make policies explicit
- Implement feedback loops
- Applycontinuous and incremental improvement

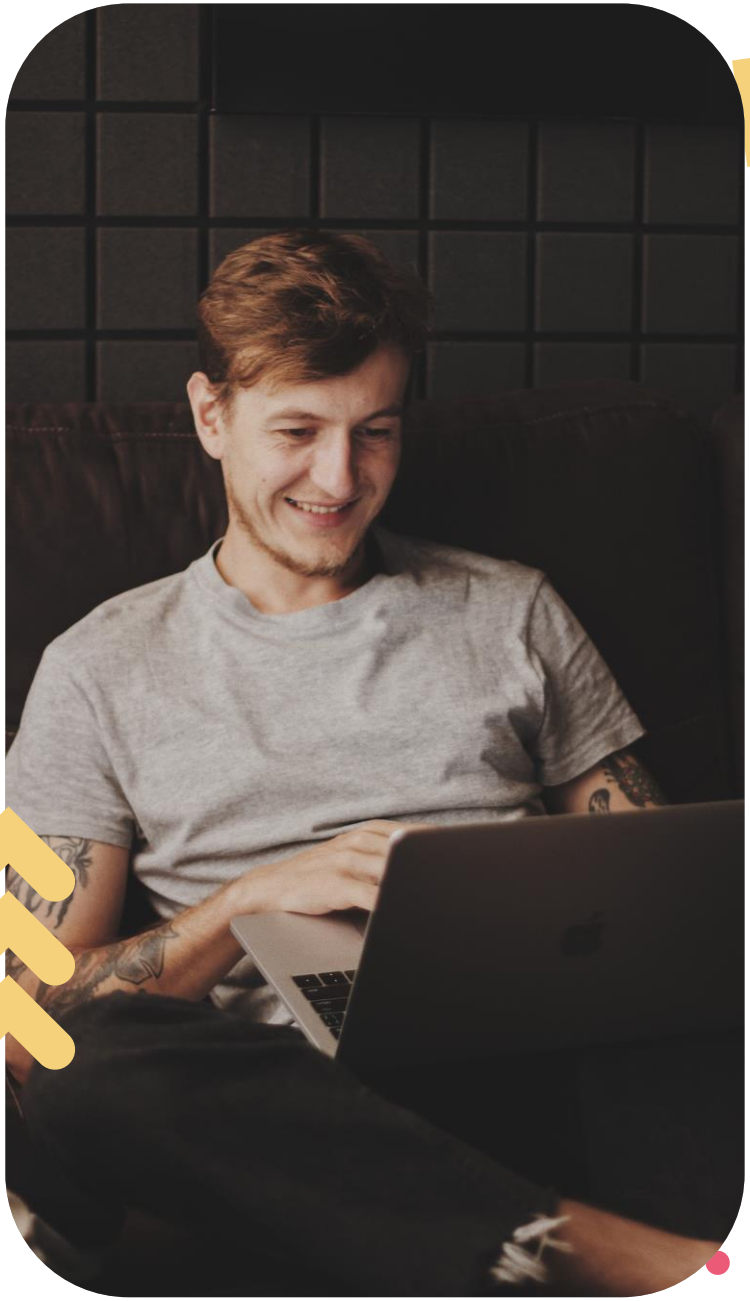
Kanban roles

- Adopting Kanban has no specific roles
- Service request manager
- Service delivery manager

Kanban lifecycle

Feedback loops are:

- Strategy review
- Operations review
- Risk review
- Service delivery review
- Replenishment meeting
- Kanban meeting stand-up
- Delivery planning meeting



Other models

DevOps

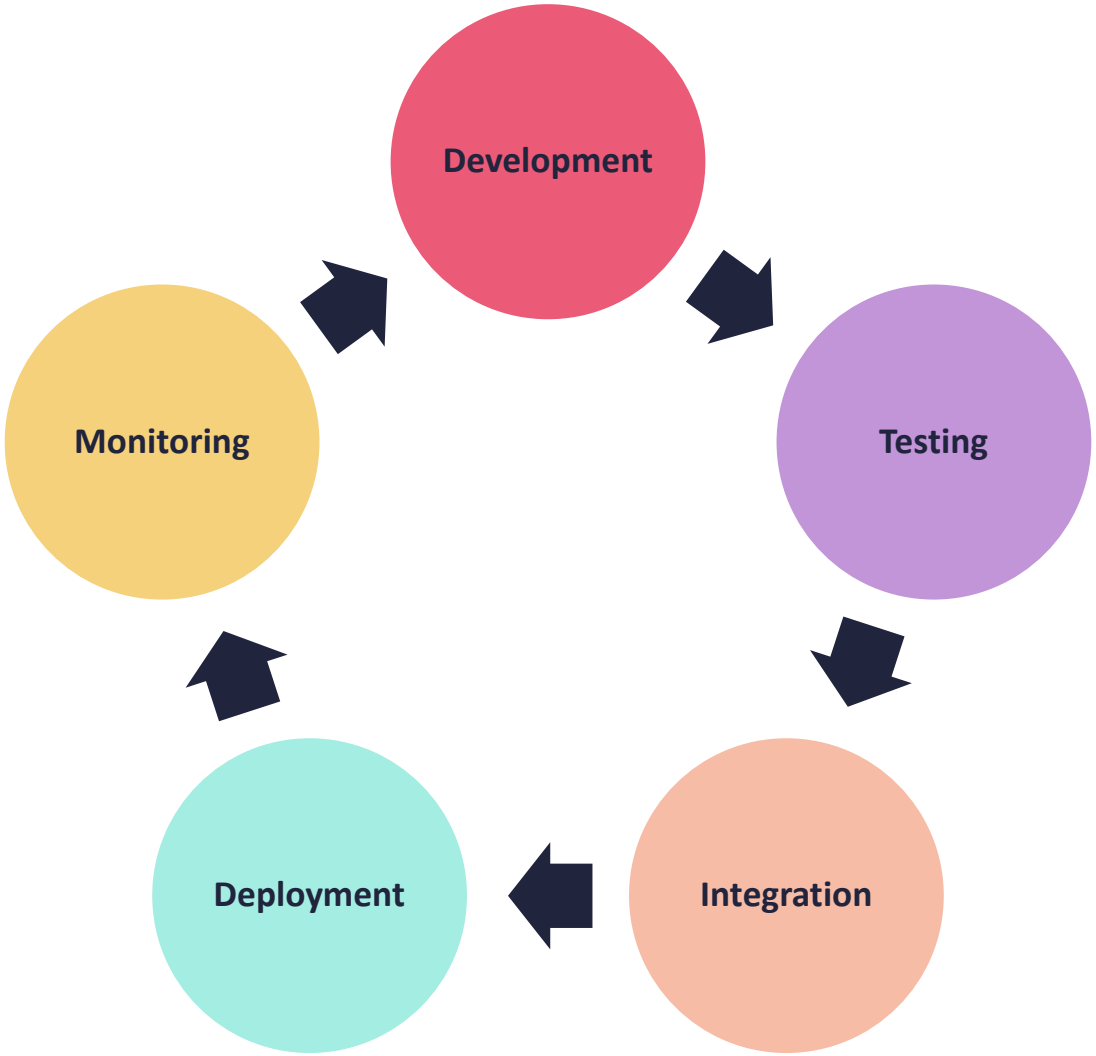
DevOps principles

- Customer-centric action
- End-to-end responsibility
- Continuous improvement
- Automate everything
- Work as one team
- Monitor and test everything

DevOps ideology

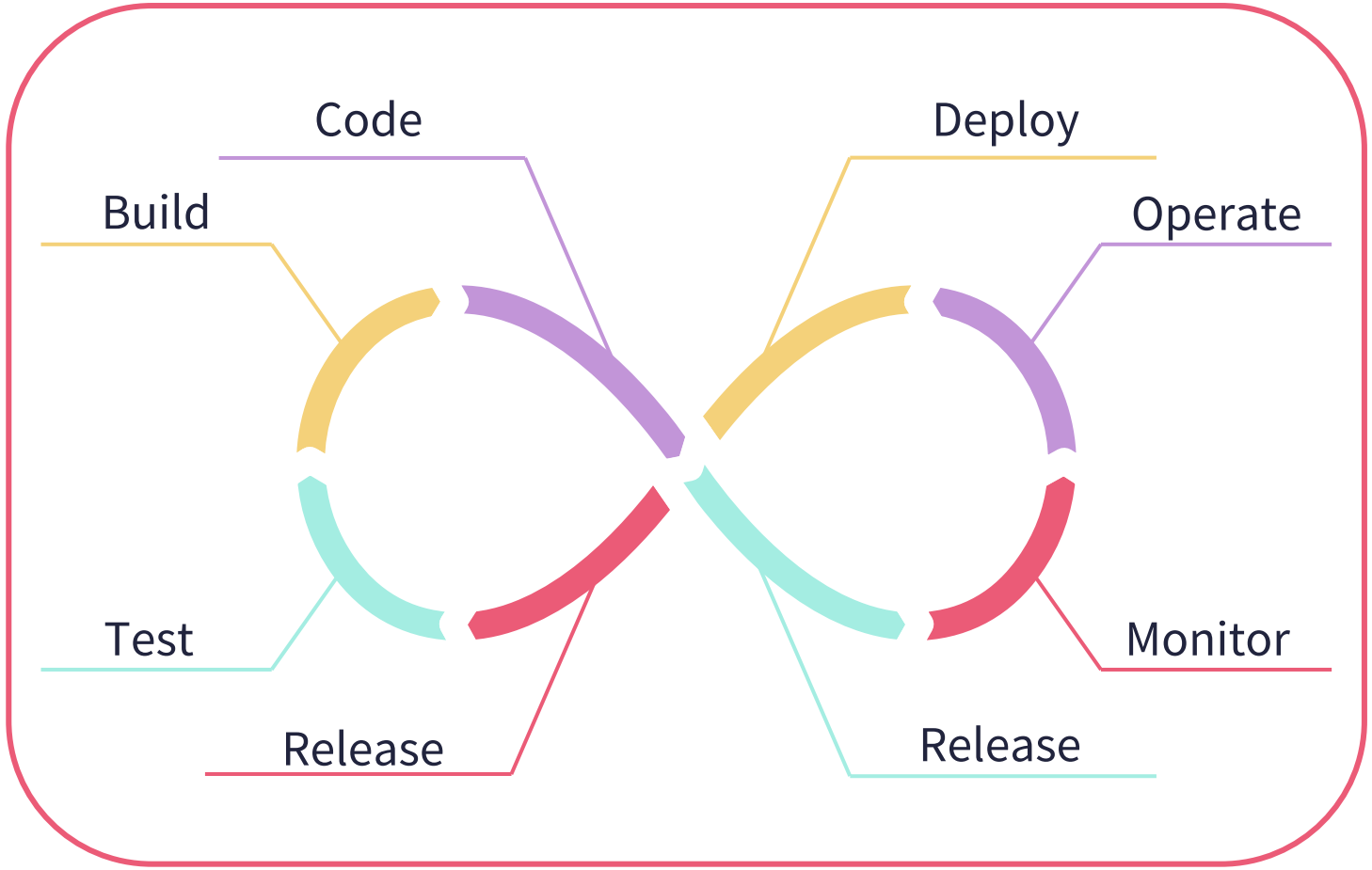
- Predictability
- Reproducibility
- Maintainability
- Time to market
- Greater quality
- Reduced risk
- Resiliency
- Cost efficiency
- Breaks larger code base into small pieces

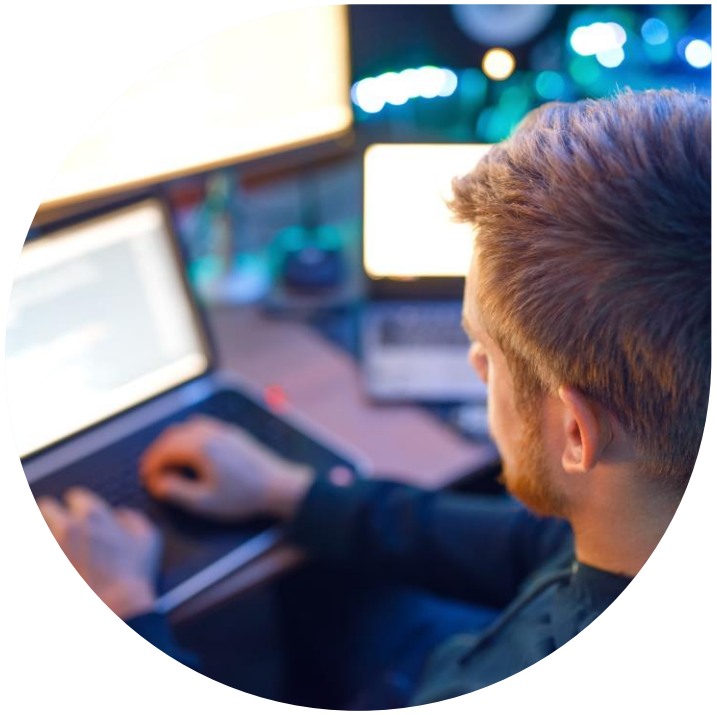
DevOps lifecycle



DevOps workflow

- Provide a visual overview of the input
- Allows for the ability to separate and arrange jobs





Feature-driven development (FDD)

Key features

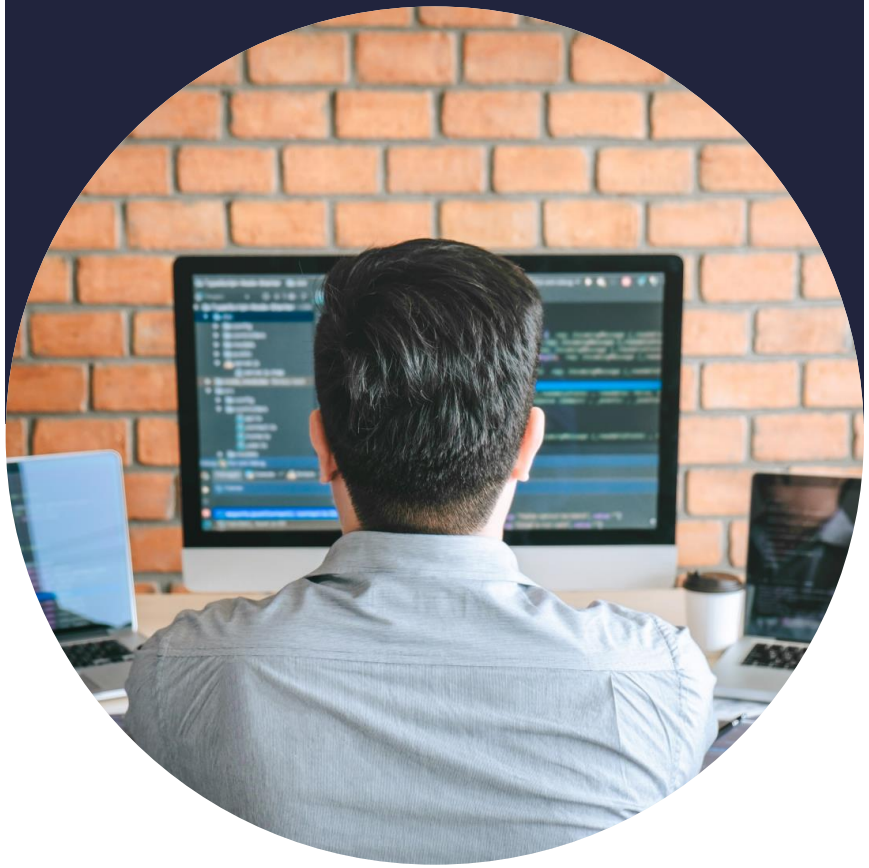
- Highly and short iterative
- Emphasises quality at all steps
- Delivers frequent, tangible working results at all steps
- Provides accurate and meaningful progress and status information

Feature-driven development

- Decomposes the entire problem domain into tiny problems
- Decomposed problems are independent from each other
- The concept of quality is broadened

Project manager

- Administrative head of the project
- Reports progress, managing budgets, resources, etc.

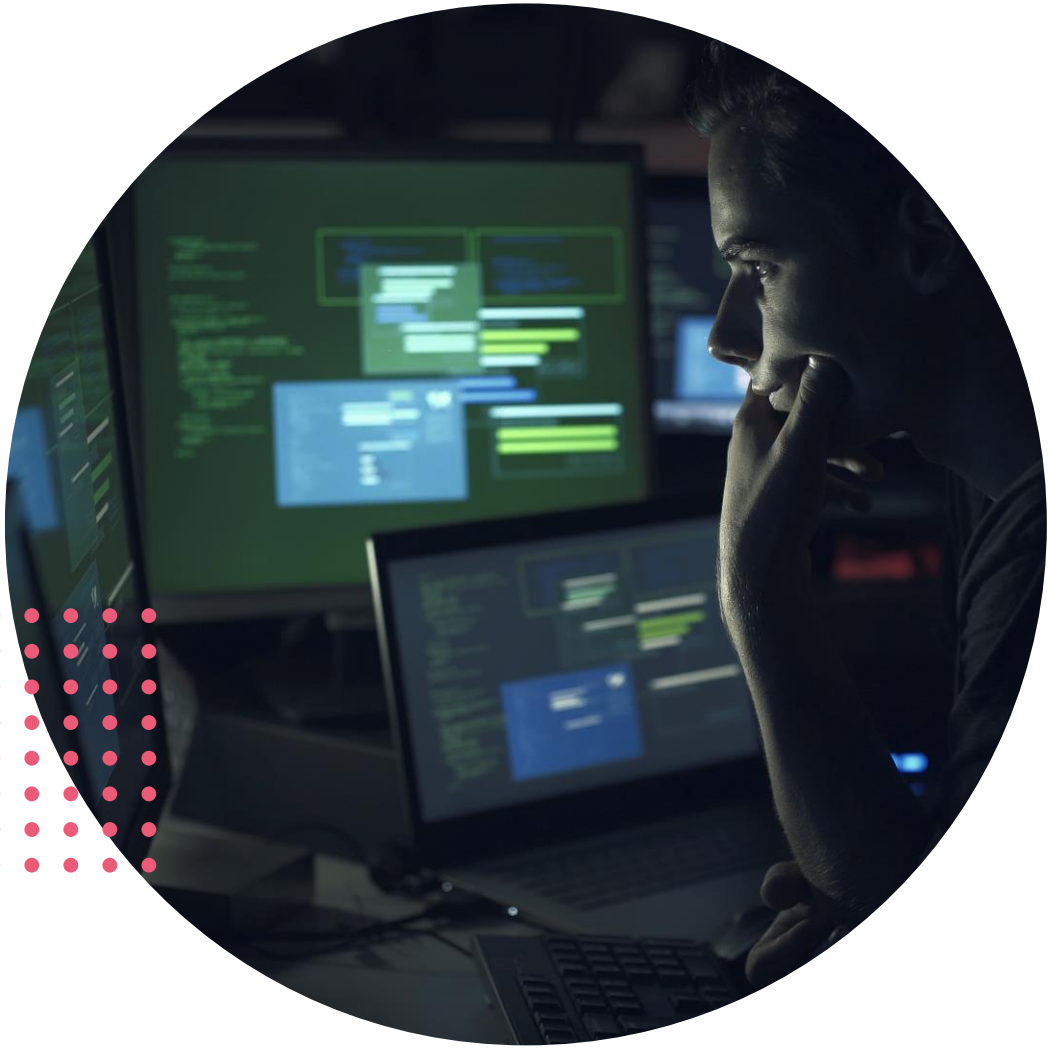


Chief architect

- Overall design of the system
- Steers the project through the technical obstacles confronting the project

Development manager

- Leads the day-to-day development activities
- Resolves everyday conflicts for resources

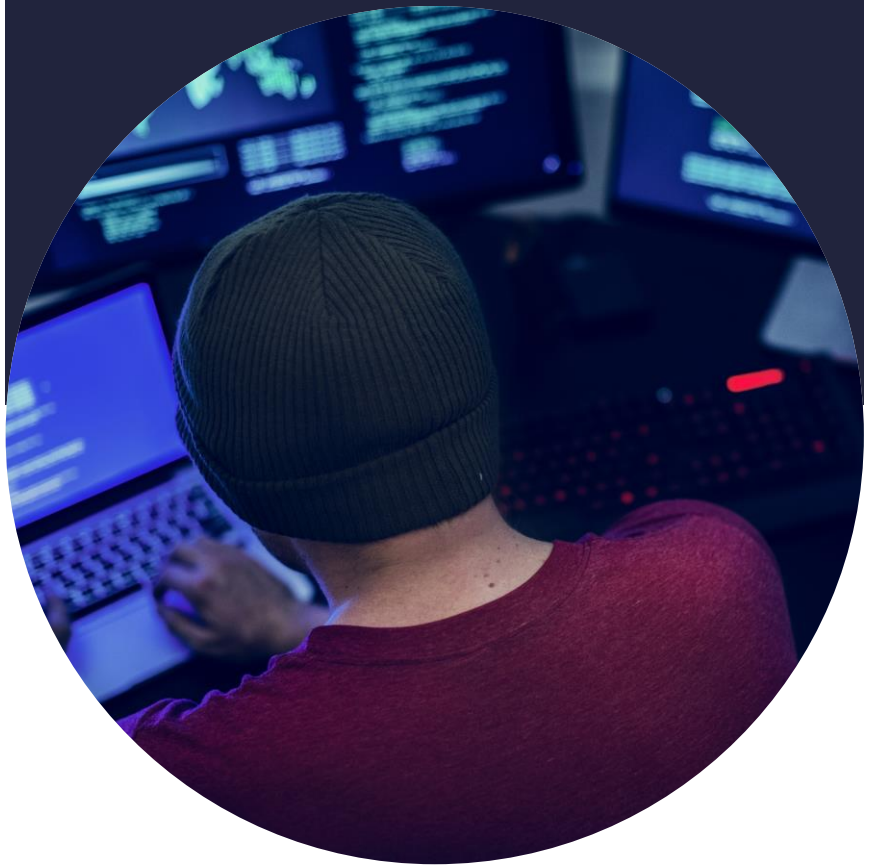


Chief programmers

- Experienced developers who have been through the entire software development lifecycle a few times
- Participate in the high-level requirements analysis
- Design activities of the project and are responsible for leading small teams

Did you know?

The name ‘Scrum’ began as a term in rugby. The term basically describes a team that organises themselves and begins moving down the field together, as one.

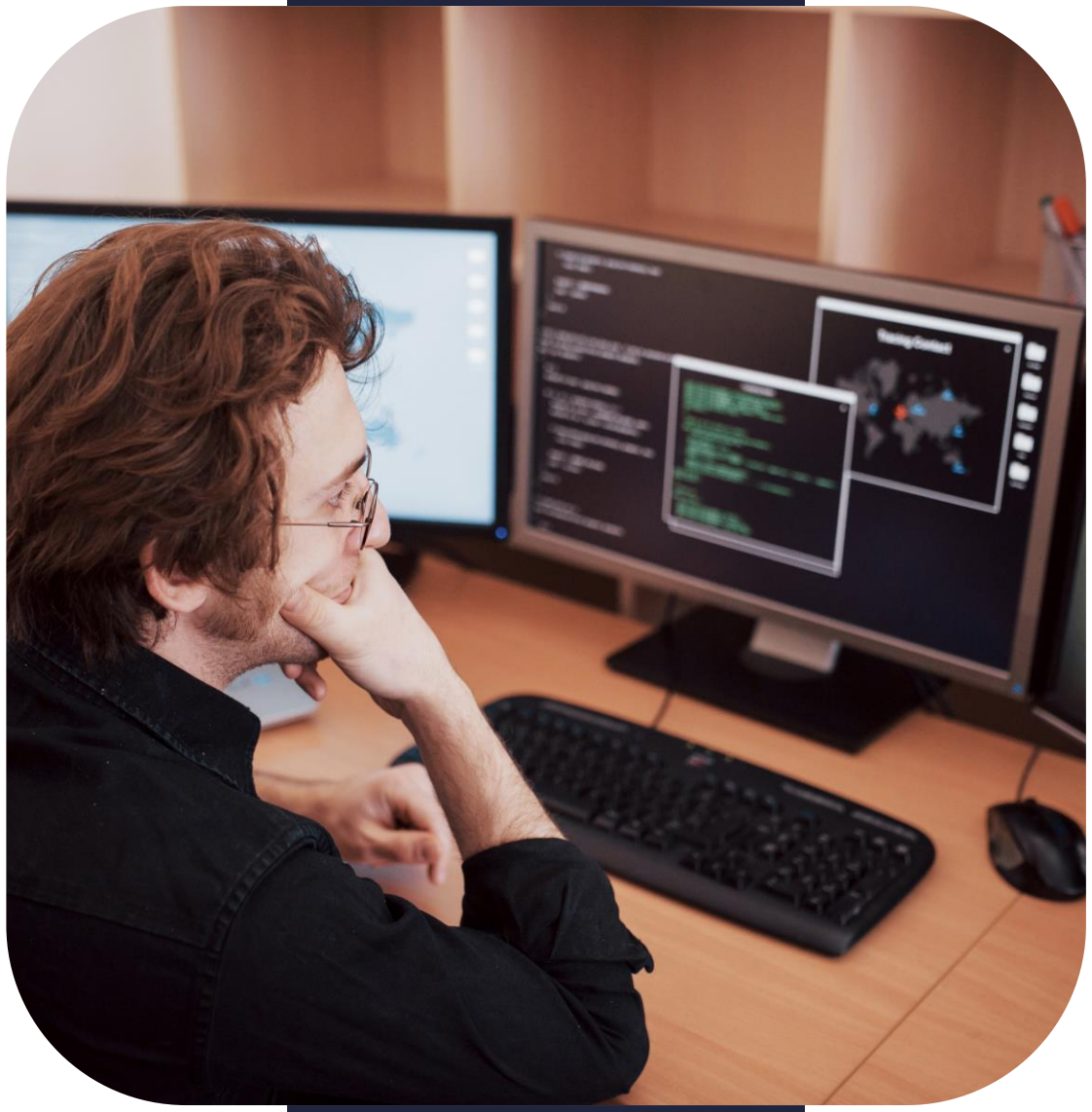


Class owners

- Developers who work as members of small development teams under the guidance of a chief programmer
- Design, code, test, and document the features

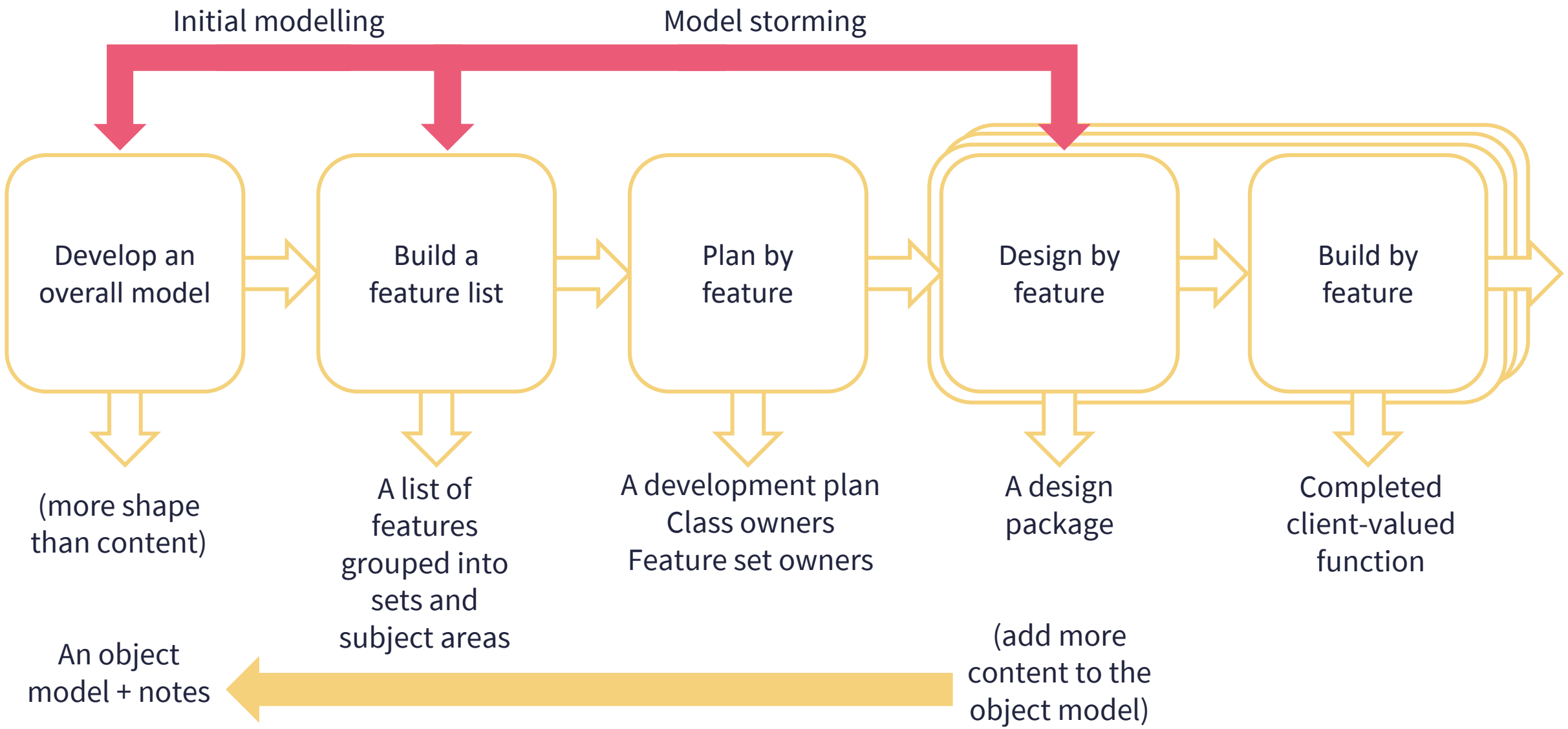
Domain experts

- Users, sponsors, business analysts
- Knowledge base that the developers rely on



Supporting roles

- Domain manager
- Release manager
- Language guru
- Build engineer
- Tool smith
- System administrator
- Testers
- Developers
- Technical writers


Initial modelling
Model storming
Develop an overall model
Build a feature list
Plan by feature
Design by feature
Build by feature
(more shape than content)
A list of features grouped into sets and subject areas
A development plan
Class owners
Feature set owners
A design package
Completed client-valued function
An object model + notes
(add more content to the object model)

FDD steps

- Domain object modelling
- Regular builds (similar to XP)
- Configuration management
- Reporting/visibility of results

Challenge

Identify the best software development strategy that we can use in our project.