1 Software process models: waterfall

1.1 Koncept

1.1.1 Software engenering

Man begyndte at bruge Software engenering, da der var stor kompleksitet ved projekterne og der skete mange fejl.
Man kigger på

- Specifikation
- Design og implementering
- test og validation
- Evolution – changing the system in response to changing customer needs

Software process model: Abstrakt repræsentation af process

1.1.2 Waterfall

Det er en plandriven model, hvilket betyder man starter ud med at ligge hele planen og den følger man så uden at kigge tilbage.
Man starter ud med at lave krav, som man så ikke ændre senere, hvor efter man går videre og design fasen, konstruktion, integration, test, installation og så til sidst vedligholdelse.

Man bør bruge den hvis man arbejder med embedded systems, livskritiske systemer eller meget store systemer, da det her kan være svært at lave det samarbejde, som de agile modeller kræver.

Den kræver kun man har kunden indover ved skift mellem phaser.
Det er godt hvis ens kontrakt limiter kravene fra starten af.
Vi bruger en *Do everything we agreed on* tilgang, da det mindser risikoen for fejl og kunden får alt hvad de har bedt om.
Der er meget lidt samarbejde i teamet ud over ved aflevering, da alle opgaver er sat.
Prisen er gerne bestemt på forhånd, da man allerede ved hvad der skal ske.

1.2 pro/con

Waterfall er rigtigt godt hvis man har en kontrakt der meget specifikt beskriver hvad der er for nogle krav, som man skal igennem.
Den fungere ikke hvis man har et projekt hvor der er konstant skiftende krav, da den ikke tillader man går tilbage og ændre i krav.
Korte projekter er ofte lette er overskue fra starten af, så der er ikke nogen grund til at bruge den ekstra energi på en agil model.

Man sætter sig ud for hvad ens mål er fra starten af, så alle ved hvor det er man vil hen.
Der bliver dokumenteret rigtigt godt, da der er mulighed for at det er forskellige teams, som står for de enkelte steps.
Man har sjældent kunde/end user med ind over igennem processen, så der er mulighed for at man ender ud med noget andet end det de vil have, da de sjældent ved hvad det er de vil have.

1.3 Andre emner

Software process model: incremental and iterative
Software process model: integration and configuration
Kombinering

2 Software process model: incremental and iterative

2.1 Koncept

Software engenering

Man begyndte at bruge Software engenering, da der var stor kompleksitet ved projekterne og der skete mange fejl.
Man kigger på

- Specifikation
- Design og implementering
- test og validation
- Evolution – changing the system in response to changing customer needs

Software process model: Abstrakt repræsentation af process

Inkrementiel / Iterativ

Man kombinere krav, udvikling og validering.
Det kan både være plandriven, agilt og oftest et mix.
Man udvikler en lille del af produktet (fx UI) får så feedback hvor efter man kan gå videre med den næste del.

2.2 pro/con

Man kan bruge både plan-baseret og agil, hvilket giver mulighed for at bruge den model, som man føler sig mest tryk med.
Det er let at få feedback.
Der er mulighed for at aflevere brugbar software tidligere, da man hele tiden fokusere på enkelte fonktioner.
En manager har brug for at få regulær updates, da det er meget svært at overværer processen.
Der kommer let noget rodet koden, som increments bliver tilføjet.

2.3 Andre emner

Software process model: Waterfall
Software process model: integration and configuration
Kombinering

3 Software process model: integration and configuration

3.1 Koncept

Software engenering

Man begyndte at bruge Software engenering, da der var stor kompleksitet ved projekterne og der skete mange fejl.
Man kigger på

- Specifikation
- Design og implementering
- test og validation
- Evolution – changing the system in response to changing customer needs

Software process model: Abstrakt repræsentation af process

Integration and configuration

Man kigger meget på software, som allerede er blevet udviklet og om man kan genbruge noget af det.
Når der bliver kodet noget nyt sørges der for at der er mulighed for at genbruge det ved at configurer det, da det giver mulighed for at sparer tid/penge.

3.2 pro/con

Da man genbruger ting bliver der en mindre risiko for at ting ikke virker, men folk får ikke når så meget en ejer følelse af det hvilket kan sænke kvaliteten af det generelle produkt.
Der er en risiko for at man bliver nødt til at gå på kompromi med krav hvis man ønsker at bruge allerede udviklet software, som ikke lever helt op til kravene.

3.3 Andre emner

Software process model: Waterfall
Software process model: incremental and iterative

4 Comparison of plan-driven and agile software engineering processes, including analysis of home grounds

4.1 Koncept

Plandriven tror på at et resultat kan blive forudset, hvor et agilt forventer at der kommer ændringer og tager højde for dette for at det skal give den bedste value.
CI clean build af systemet flere gange om dagen. I det agile builder man mange gange for at se om det virker.
Prototype lave krav sammen med kunde. Prototype ved agile kan vises som et færdigt produkt efter hver iritation og så udvikle på den så det kan ses hvilken retning kunden vil.
XP practices Pair programming, planning game, TDD
Scrum roles Product Owner, ScrumMaster, Development Team
Practices collaborative, Open communication, adaptation, trust, Sprint Planning+Daily Scrum+Sprint Review+Sprint Retrospective,+Baclog refinement.
Home ground

Plan-driven	Agile
Plan orientede udviklere (mix af skill)	Velvidende, samlet og samarbejede udviklere
Mix af kunde kababilitet nuveauer	højt empowered kunder
God dokumentation	Forventer ofte ændringer
Krav kendt tidligt og forsøgt at set kommende	Ændringer er billige
At lave om er dyrt	Mindre teams
Store teams	Hurtig "value"
Høj sikring	•

4.2 pro/con

Agile er til små
Plan-driven er til store
Mix

4.3 Andre emner

Software process model: Waterfall
Software process model: incremental and iterative
integration and configuration

5 Key features of Scrum

5.1 Koncept

A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.

- Roles
  Product Owner: Ejeren af produktet, derfor ikke en del af teamet, men hjælper med at sætte krav.
  Scrum Master: Styrer teamet i forhold til hvad der skal ske hvornår, er ikke en del af teamet.
  Team: Programører, designere mm.
- **Sprint Planning**: Planlægning.
- **Daily Scrum**: Dagligt møde.
- **Sprint Review**: Review afholdt ved afslutningen af hvert sprint.
- **Sprint Retrospective**: Møde omkring hvad gik godt, hvad kan gøres bedre og hvad vil vi gøre for at gøre det bedre.
- **Backlog refinement**: Holde backlog opdateret.
- **Product Burndown/Sprint Burndown**: Completed work per day for det nuværende projekt udgivelse.
- **Scrum board**: Fremvisning af backlog

Scrum giver **commitment**, **courage**, **focus**, **openhed** (team) **respect** (management over for team, derfor tror de på de gør deres arbejde)

Det sker nogen gange følgende **fejl**: Scrum master implemented as manager who tells team what to do (right way: Facilitator for team), kunde bliver ikke involveret, nye krav og opgaver bliver tilføjet i løbet af et sprint.

eXtreame Programing er godt ved scrum i forhold til kunde-on-site, user stories, planning game.

5.2 pro/con

Effektivt brug af tid og penge
Gøre store projekter mere overskulige
Godt for hurtigt bevægende projektor
Teams for godt overskulighed
Adopts feedback
Man kan se arbejdet hver person laver
Ens scope kan let blive uroligt

Hvis folk ikke er committet går det let galt
Kan være svært at få igang i store teams
Daglige møder iritere folk

5.3 Andre emner

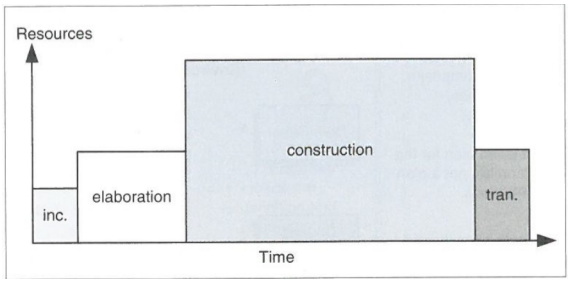
Agile

6 Key features of RUP

6.1 Koncept

Framework til UP, altså et RUP produkt.
Interativ process framework.

Udviklet i små tidsboksede iterationer.
High-value og high-risk tidligt. Forsøg med genbrug.
Dette forsikre kunder om value.
Være klar på ændringer tidligt i processen.
Team work er vigtigt.
4 phaser: Inception, elaboration, construction, transition.



Inception: Establish, project schedule and cost estimate, gennemførlig Make mussiness case, define scope.

Elaboration: Address know risk factors, establish and validate system architecture, mange system requirements, plan for the construction phase (cost og schedule)

Construction: Lave det sidste, series of short time-boxed iterations, use-case, UML

Transition: deploy, feedback for refinements, user training

Discipline

Requriment, design, test, implemtering, projekt mangement

Best practice

Timeboxed iterations, reuse, vertify quality, visual modelling, manage requirements, manage change.

Misforståelser

Iteration bliver for lang
De bliver ikke time boxed (bliver udvidet)
De ender ikke i test/integret

Scum: If the predefined optional activities of UP is seen as requried tasks it is bad, but if they are seen as optional advice it is okay.

XP:Upfront moddeling: UP uses accepts much more time
Early iteration: XP does not look for high-value and high-risk but might find them

6.2 pro/con

Pros: RUP takes the best parts of Waterfall and incorporates them into a more iterative process that allows for changes.
Cons: Like Waterfall, RUP is also process-heavy, and can rely too heavily on stakeholder feedback. Even as an iterative process it can be too slow for certain types of projects.

6.3 Andre emner

Scrum

7 Requirements Elicitation and Management

7.1 Koncept

Main activites finde og analysere needs, specificere krav, validere krav.
Organized spiral: requirement validation, risk analysis, models and prototype.
Requirements elicitation: find og forstå krav, organiser dem, prioter, dokumenter, forfra.
Techniques interview (open/closed), etnography (observer), prototype.
communicate: Stories/senaries
Dokument

Waterfall: Approved requirements document with strict change management

Scrum: Product vision and product backlog, reviewed and updated every sprint

negotiated

Waterfall: Up front in the requirements phase – state it now or it will be difficult later to get it

Scrum: Ongoing refinement of product backlog with stakeholders, say what is most important now, we will continue

7.2 pro/con

Mange kunder har krav som modgår hinanden, de snakker deres egent sprog så der sker let forvirring.

7.3 Andre emner

Configuration Management

8 Managing change to requirements

8.1 Koncept

De bliver manged for at man kan holde styr på de krav som kommer fra kunderne, så man kan se om man får klaret dem.
Man kigger også efter de versioner man laver af software, så det er muligt at gå tilbage og se hvad der er sket.
Ændringer bliver godkendt af forskellige personer alt efter modellen der bliver brugt. I en plan driven er det ofte en projekt manager eller en styregruppe og ved en agil er det kunden.
Får at holde prisen nede kan man arbejde ud fra expect changes hvor man fx laver prototyper, så kunden let kan se hvad der sker eller tolerance to changes hvor man designer så det er let at lave ændringer.
$Exposure(r) = P(r) * L(r)$

8.2 pro/con

Prototype den bliver brugt anderledes end hvordan det endelige produkt skal bruges. Den bliver ikke brugt af de "endelige" brugere. Der bliver ikke brugt nok tid på at træne brugerne i at bruge den.
incremential development
Kunder kan bruge det som prototyper som bliver vidre udviklet. De kan se værdig tidligt. Let at få ændringer ind undervejs.
Problematisk hvis det er hele systemer der skal laves ændringer i eller hvis det skal arbejde sammen med gamle systemer. Svært at definere en del base for hele systemet tidligt i udviklingen.

8.3 Andre emner

Configuration Management

9 Quality Control: Verification and Validation

9.1 Koncept

Validation: Are we building the right product?
Verification: Are we building the product right?
Inspection: Analyser, check system krav, design modeller, soruce code, test.
Test: living up to the requriments, code folows the design, models, requriments
Peer review: peers to check your code and more
TDD: Vi skriver en test og ud fra den udvikler det. Dette gør at vi ved det virker før vi går vidre og det tvinger os til at skrive tests hvilket ellers godt kan være noget som man kommer til at hoppe over.
regressiontest re-running previous tests
agile practices Definition of Done, Sprint Review, Check before check-in, Never break the build, Fix problems when you see them, Culture, XP: Customer on site, XP: Pair programming

9.2 pro/con

It can help to prevent faulty goods and servies being sold
It does not prevent waste of resources when products are faulty

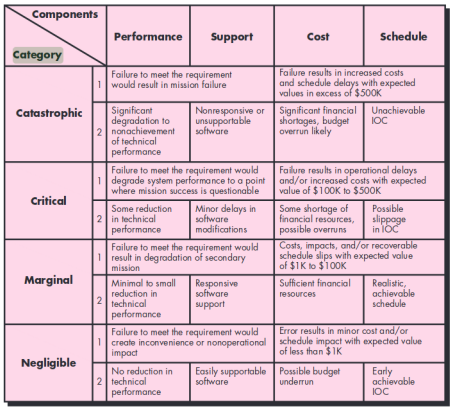
9.3 Andre emner

Quality Management

10 Risk Management

10.1 Koncept

Something that may happen and causes a loss
Kategorier Catastrophic, Critical, Marginal, Negligible, Uncertainty, project, technical, business
- Keyperson from team dies, a supplier is not delivering as promissed



Components / Category		Performance	Support	Cost	Schedule
Catastrophic	1	Failure to meet the requirement would result in mission failure		Failure results in increased costs and schedule delays with expected values in excess of $500K	
	2	Significant degradation to nonachievement of technical performance	Nonresponsive or unsupportable software	Significant financial shortages, budget overrun likely	Unachievable IOC
Critical	1	Failure to meet the requirement would degrade system performance to a point where mission success is questionable		Failure results in operational delays and/or increased costs with expected value of $100K to $500K	
	2	Some reduction in technical performance	Minor delays in software modifications	Some shortage of financial resources, possible overruns	Possible slippage in IOC
Marginal	1	Failure to meet the requirement would result in degradation of secondary mission		Costs, impacts, and/or recoverable schedule slips with expected value of $1K to $100K	
	2	Minimal to small reduction in technical performance	Responsive software support	Sufficient financial resources	Realistic, achievable schedule
Negligible	1	Failure to meet the requirement would create inconvenience or nonoperational impact		Error results in minor cost and/or schedule impact with expected value of less than $1K	
	2	No reduction in technical performance	Easily supportable software	Possible budget underrun	Early achievable IOC

how

- Identifiser, udregn exposure, beskriv
- Priotiser
- RMMM

Waterfall / plan driven en del af planerne, del af andre planer, det bliver planlagt hvordan.
agile Daily Scrum, sprint review, sprint retrospective.
spiral risk driven, starter hver iteration ud med risk, og gør det tidligt

10.2 pro/con

10.3 Andre emner

11 Project Planning and Management

11.1 Koncept

plandriven project planning plan arbejdet, gør det, antag du kan forudse hvad der er leveret
Progress Milepæle og dokumentation
Agil planning Ændringer, prioritiseret backlog
Estimation Taget ud fra forud viden, algorime baseret (modeller og parameter)
Agile estimation Velocity af story points
sprint planning planlægning sammen med scrummaster, productowner og teamet
Planning game i XP For hver opgave giver man et estimat på hvor lang tid det ca tager. Kortest vs længest diskuterer hvorfor de synes det.

11.2 pro/con

11.3 Andre emner

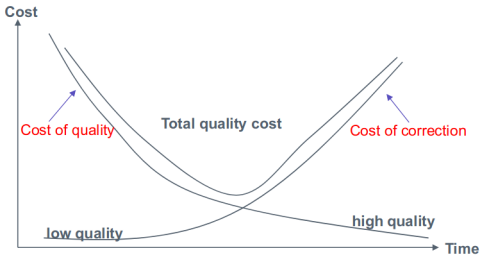
12 Quality Management: How is quality defined - agile versus plan driven approaches

12.1 Koncept

Forskellen mellem plandreven og agilt er at plandreven går ud fra at man kan forudsige resultatet, og agilt forventer at ændringer kommer til at ske. Hvordan definerer Böhm / Turner de **primære faktorer**?

- Applikation er små, hurtige ændringer, turbulent environment
- Management onsite, kvalitetskontrol, stilletiende viden
- Teknisk: Prioriterer uformelle requirements og simplet design
- Mennesker: Cockburn L2 og L3 udviklere

Grunden til at requirements ændrer sig kan være forretningsmæssige, teknologiske eller fordi de lærer at bruge det.
Hvis at noget skal ændres, så kan man ændre process og analysere impakten på ændringen



Trade of er at man vælger noget fram for noget andet fx man tester mindre for at blive hurtigere færdig.
Quality culture alle føler sig ansvarlige for softwaret
Agile Meget relesable code, unscheduled product reviews, schilude reviews, status møder, mange tests.
Pair programming to programører skriver sammen, man kan skrive tests samme tidigt med koden skrives, fejl findes tidligt, det er god kode deling til hvis en fx stopper.
Validation: Are we building the right product?
Verification: Are we building the product right?

12.2 pro/con

Quality Management assumes good process good –¿ product, or same process delivers same quality

12.3 Andre emner

13 Configuration Management

13.1 Koncept

Disciplin som giver en måde til at identificere og controllere ting. Det giver en måde at håndtere integritet og kvalitet.
Branching fordele opgaver ud i forskellige grene, så man ikke arbejder i de samme filer.
Merging sætte ændringer sammen fx fra grene.
VC system
Centralize: En fælles kopi som alle sender til
Distributed: Alle har meta data, kan gøre det samme som centralized. Hvis man har mange ændringer eller mange binærer filer kan det komme til at fylde rigtigt meget
Release: ALT!
Continous integration: Agile teams typically configure CI to include automated compilation, unit test execution, and source control integration. Sometimes CI also includes automatically running automated acceptance tests.
Daily build: Performing daily builds helps ensure that developers can work knowing with reasonable certainty that any new bugs that show up are a result of their own work done within the last day.

13.2 pro/con

Plandriven
Strong emphasis on testing
Validation of internal and external deliverables not just the product.
Definition of requirements before designing the system.
Enable project management to track the progress very accurately. The completion of each phase is a milestone in the project plan.
Does not handle concurrent events.
Does not handle iterations.
Can't handle the dynamic changing of requirements during the lifecycle.
Agile
Not optimized for sprint Slow for testing
easy share
easy release

13.3 Andre emner

14 Andet

14.1 Scrum

Roles
Product Owner: Ejeren af produktet, derfor ikke en del af teamet, men hjælper med at sætte krav.
Scrum Master: Styrer teamet i forhold til hvad der skal ske hvornår, er ikke en del af teamet.
Team: Programører, designere mm.
Sprint Planning: Planlægning.
Daily Scrum: Dagligt møde.
Sprint Review: Review afholdt ved afslutningen af hvert sprint.
Sprint Retrospective: Møde omkring hvad gik godt, hvad kan gøres bedre og hvad vil vi gøre for at gøre det bedre.
Backlog refinement: Holde backlog opdateret.
Product Burndown/Sprint Burndown: Completed work per day for det nuværende projekt udgivelse.
Scrum board: Fremvisning af backlog
Scrum giver commitment, courage, focus, openhed (team) respect (management over for team, derfor tror de på de gør deres arbejde)

14.2 Continuous integration (CI)

Samling af alt minimum 1 gang om dagen i XP mange gange om dagen.

14.3 Prototype

Prototype lave krav sammen med kunde. Prototype ved agile kan vises som et færdigt produkt efter hver iritation og så udvikle på den så det kan ses hvilken retning kunden vil.
Får at holde prisen nede kan man arbejde ud fra expect changes hvor man fx laver prototyper, så kunden let kan se hvad der sker eller tolerance to changes hvor man designer så det er let at lave ændringer.

14.4 Home ground

•	Agile
Prmær mål	Hurtig værdi, god til ændring
Størelse	Mindre teams og projektere
Envioment	Turbulent, høj chance, projekt dokuseret
Kunde relation	On-site kunder, fokus på increments
Plan/kontrol	Kvalitet over kvuantitet
Kommunikation	Interpersonal knowledge
Krav	priotiserede informative historier og test cases, kan klarer uforusete ændringer
Udvikling	Simpelt design, små increments, refactor billigt
Test	udførbarer test cases definere krav
Kunde	Dedicted
Udviklere	ingen under level 2 cockburn

•	Plan-driven
Prmær mål	Forventning, stabil, høj sikkerhed
Størelse	Støre teams og projektere
Envioment	Stabilt, lav ændring, projekt/organzition fokus
Kunde relation	As-needed customer, kontrakt, evolutionary
Plan/kontrol Kvuantitet og kvalitet	
Kommunikation	Dokumenteret knowledge
Krav	Formaliseret projekt, interface qualitet, forudser krav
Udvikling	Akitecht paralel udvikling, længre incrementer, refactor er d
Test	Dokumenteret test planer og procedyrer
Kunde	ikke altid sammen
Udviklere	Level 3 tidigt efter følgende få; 30 1B

14.5 Requirement/ændringer

Krav er stillet af kunden til noget produktet skal kunne når det er færdig udviklet
Ved plan-driven stilles de alle sammen til at starte med.
Ved agile de stilles under vejs, da der tages højde for der kan komme ændringer.

14.6 Cockburn

Level 3: Ændre en metode (Bryde regler) til at passe til en ny ukendt situation.
Level 2: Små ændringer for at passe til en kendt ny situation
Level 1A: Kan lave metode step med mere advancerede modeller
Level 1B: Kan lave metode step med simple modeller
Level -1: Kan have tekniske skills, men kan ikke eller vil ikke samarbejde eller bruge en metode.

14.7 Dokumentation

Plan-driven kræver god dokumentation, da der let kan ske udskiftning imellem de forskelige faser fx mellem design og udvikling.

14.8 Negotiated

Waterfall: Up front in the requirements phase – state it now or it will be difficult later to get it.
Scrum: Ongoing refinement of product backlog with stakeholders, say what is most important now, we will continue.

14.9 Validation

Are we building the right product?

14.10 Verification

Are we building the product right?

14.11 Inspection

Analyser, check system krav, design modeller, soruce code, test.

14.12 Test

unit, auto, manuel
living up to the requriments, code folows the design, models, requri- ments
regressiontest re-running previous tests
Pair programming

14.13 Test Driven Development (TDD)

Test skrives inden koden skrives som der skal udviles ny funktion. Man går ikke videre indtil det virker.
Code coverage. Regression testing. Simpel debug. System dokumentation.

14.14 Risk Mitigation, Monitoring, and Management Plan (RMMM)

A risk management strategy can be included in the software project plan, or the risk management steps can be organized into a separate risk mitigation, monitoring, and management plan. The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan.

14.15 risk information sheet (RIS)

Dokumentet som informere om den enkelte risk. Ofte brugt database system til at kunne lave ændringer.

14.16 Dimensions

Personal
Dynamism: Krav ændringer / måned
Kultur: chaos vs order
Antal folk
Criticality: Tab i forhold til fejl.