SOFTWARE ENGINEERING
LECTURE 3

CARSTEN RUSENG JAKOBSEN

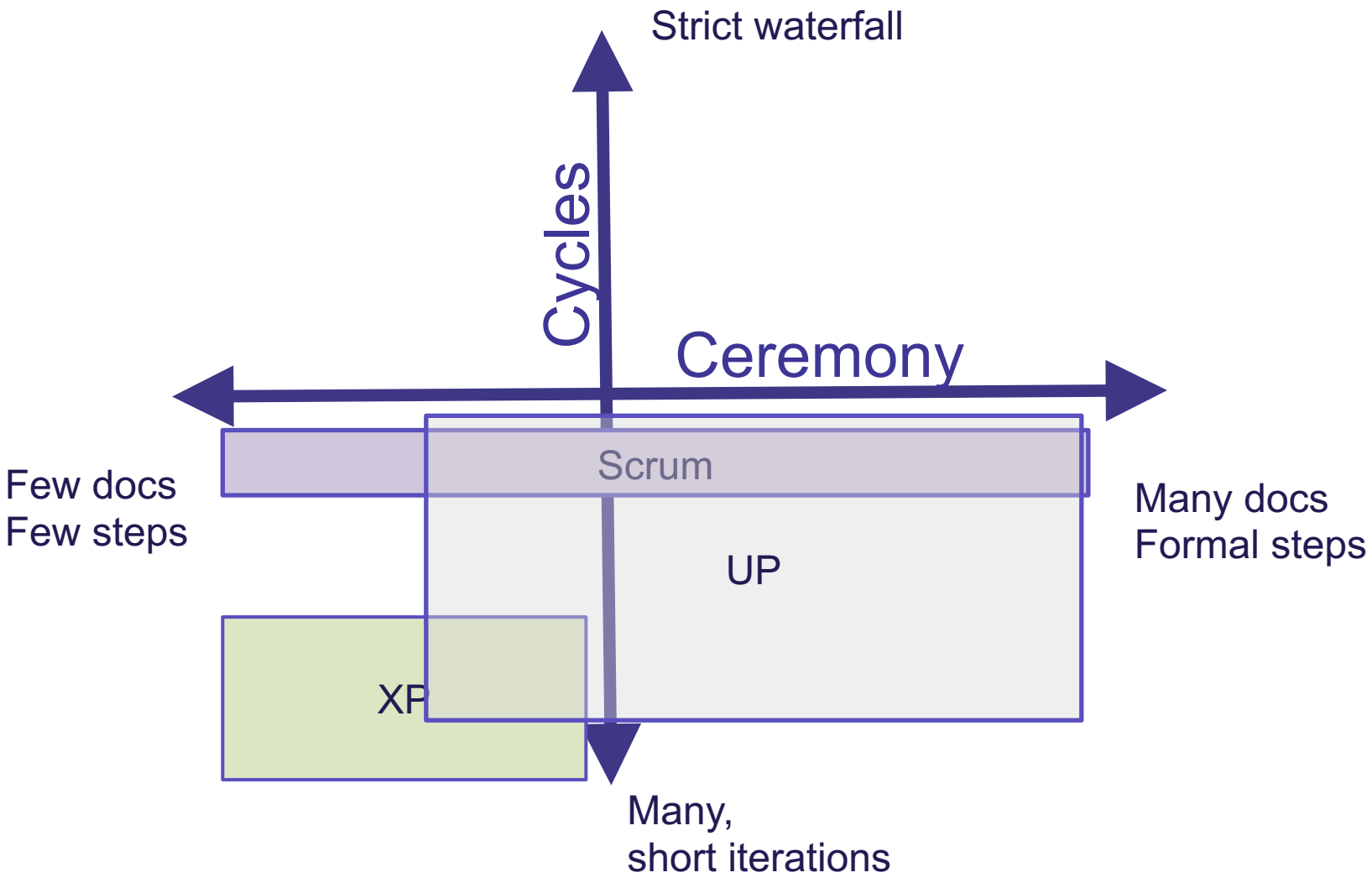
AALBORG UNIVERSITY
DENMARK

Miniproject

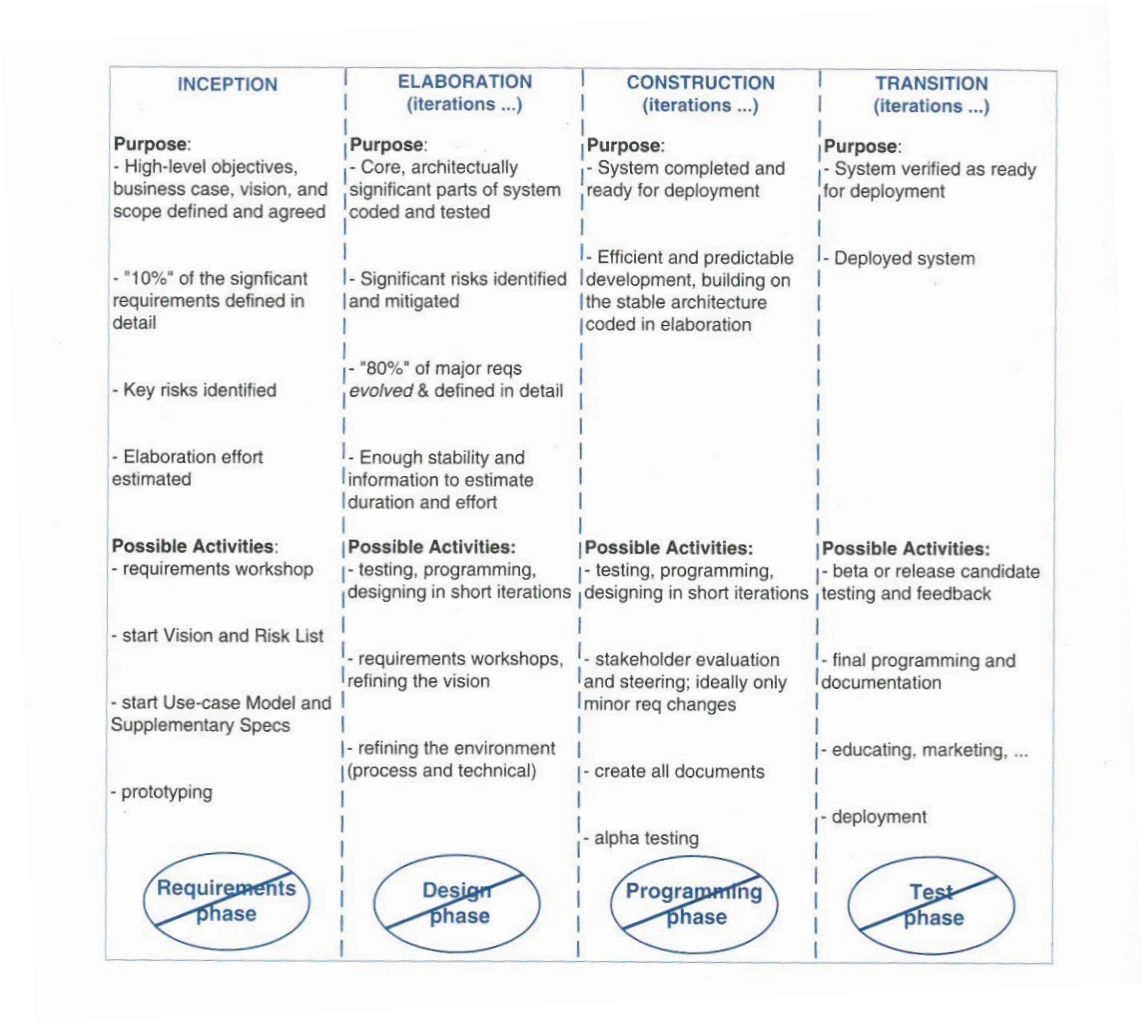
- Prerequisite for exam
- You MUST form groups of 3-5 people (not 1,2, or 6 or more)
- 3 interim deliveries, Name and room must be on frontpage

- Part 1 due at Lecture 04
  - Basis – Describe project,
  - Analysis of 3-4 methods and selection with argument, including mix
- Part 2 due at Lecture 08
  - Plan project according to choosen model and methods
  - Establish timetable, risk analysis, configuration plan, overall plan
- Part 3 due at Lecture 12
  - Evaluate practice
  - What went good/bad, as expected vs unexpected, reflection on why

Method overview

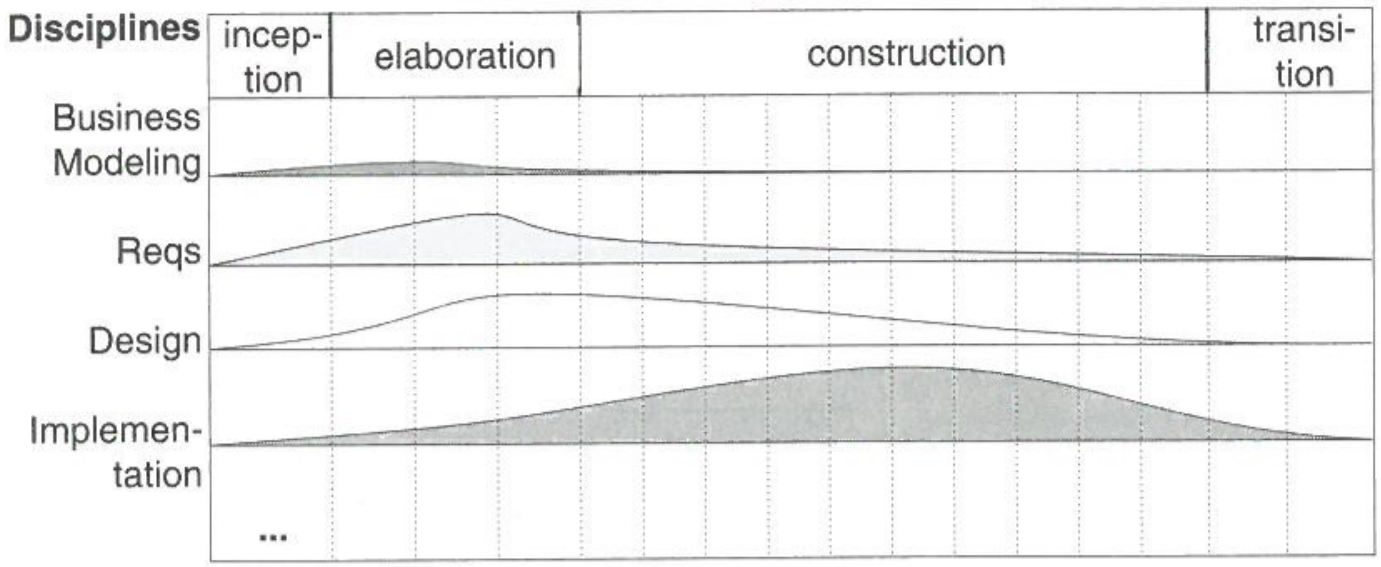


UP – not waterfall

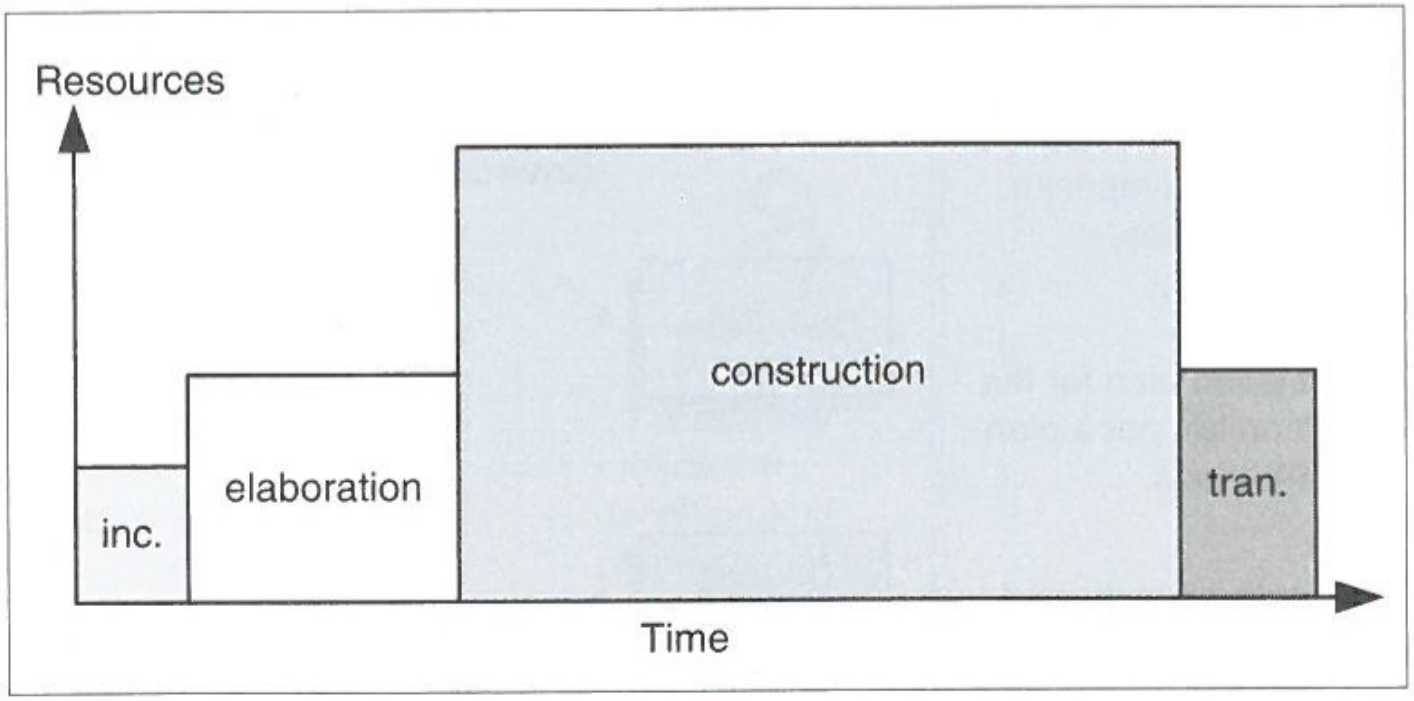


INCEPTION	ELABORATION (iterations ...)	CONSTRUCTION (iterations ...)	TRANSITION (iterations ...)
Purpose: - High-level objectives, business case, vision, and scope defined and agreed	**Purpose**: - Core, architectually significant parts of system coded and tested	**Purpose**: - System completed and ready for deployment	**Purpose**: - System verified as ready for deployment
- "10%" of the signficant requirements defined in detail	- Significant risks identified and mitigated	- Efficient and predictable development, building on the stable architecture coded in elaboration	- Deployed system
- Key risks identified	- "80%" of major reqs *evolved* & defined in detail		
- Elaboration effort estimated	- Enough stability and information to estimate duration and effort		
Possible Activities: - requirements workshop	**Possible Activities:** - testing, programming, designing in short iterations	**Possible Activities:** - testing, programming, designing in short iterations	**Possible Activities:** - beta or release candidate testing and feedback
- start Vision and Risk List	- requirements workshops, refining the vision	- stakeholder evaluation and steering; ideally only minor req changes	- final programming and documentation
- start Use-case Model and Supplementary Specs	- refining the environment (process and technical)	- create all documents	- educating, marketing, ...
- prototyping		- alpha testing	- deployment
~~Requirements phase~~	~~Design phase~~	~~Programming phase~~	~~Test phase~~

UP phases


Disciplines
incep-
tion
elaboration
construction
transi-
tion
Business
Modeling
Reqs
Design
Implemen-
tation
...

UP time and effort



UP milestones

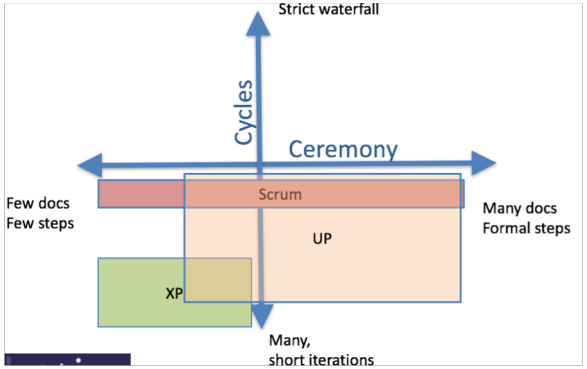
Phase	Milestone Goals	Comments
Inception	Agreement on scope, vision, and priorities. Some risks identified. A plan to start elaboration exists.	*Establish a common vision.* Typically a very short phase, such as a few days or weeks. A first requirements workshop might be held.
Elaboration	The vision, requirements, and architecture are stabilized. The core executable architecture is implemented; major risks are mitigated. The majority of requirements are defined. Estimates and coarse-grained plan are defined.	*Build and test the risky core.* This phase contains significant production programming and testing, combined with evolutionary requirements and design work. Usually composed of several iterations. In addition to programming, perhaps a series of requirements workshops; one per iteration. Semi-reliable plans and estimates at *end* of elaboration.
Construction	System is believed ready to be deployed. Stakeholders are ready for deployment.	*Build and test the rest.* Typically the largest set of iterations. As in elaboration, major testing occurs in each iteration.
Transition	System is deployed. Users are satisfied.	*Deploy.* Beta testing, release candidate evaluation, training.

UP workproducts (sample)

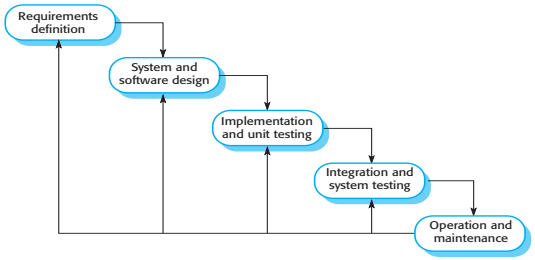
Discipline	Workproduct	Comment
Requirements	Vision	Summary of stakeholders' key needs and features.
	Use-Case Model	The set of use cases describing the intended functions and environment.
Design	Design Model	An object model describing the hardware and software realization of the use cases in terms of collaborating objects.
	Software Architecture Document	A system overview or learning aid that includes several architectural views.
Project Management	Iteration Plan	The goals and tasks for the current or next iteration.
	Risk List	A list of prioritized risks with associated mitigation plans.

Mixed approaches?

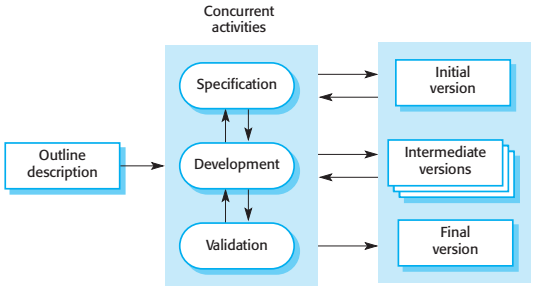
- How is UP a mixed approach?
  - Incremental and mix with agile
  - … Why? Why not?
- What happens in the different phases of UP
  - Inception



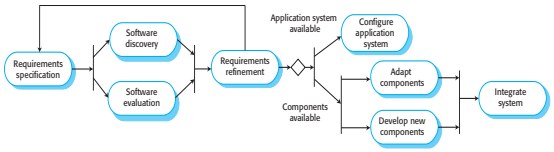
Software process models



- The waterfall model
  - Plan-driven model. Separate and distinct phases of specification, design, implementation, test and operations.



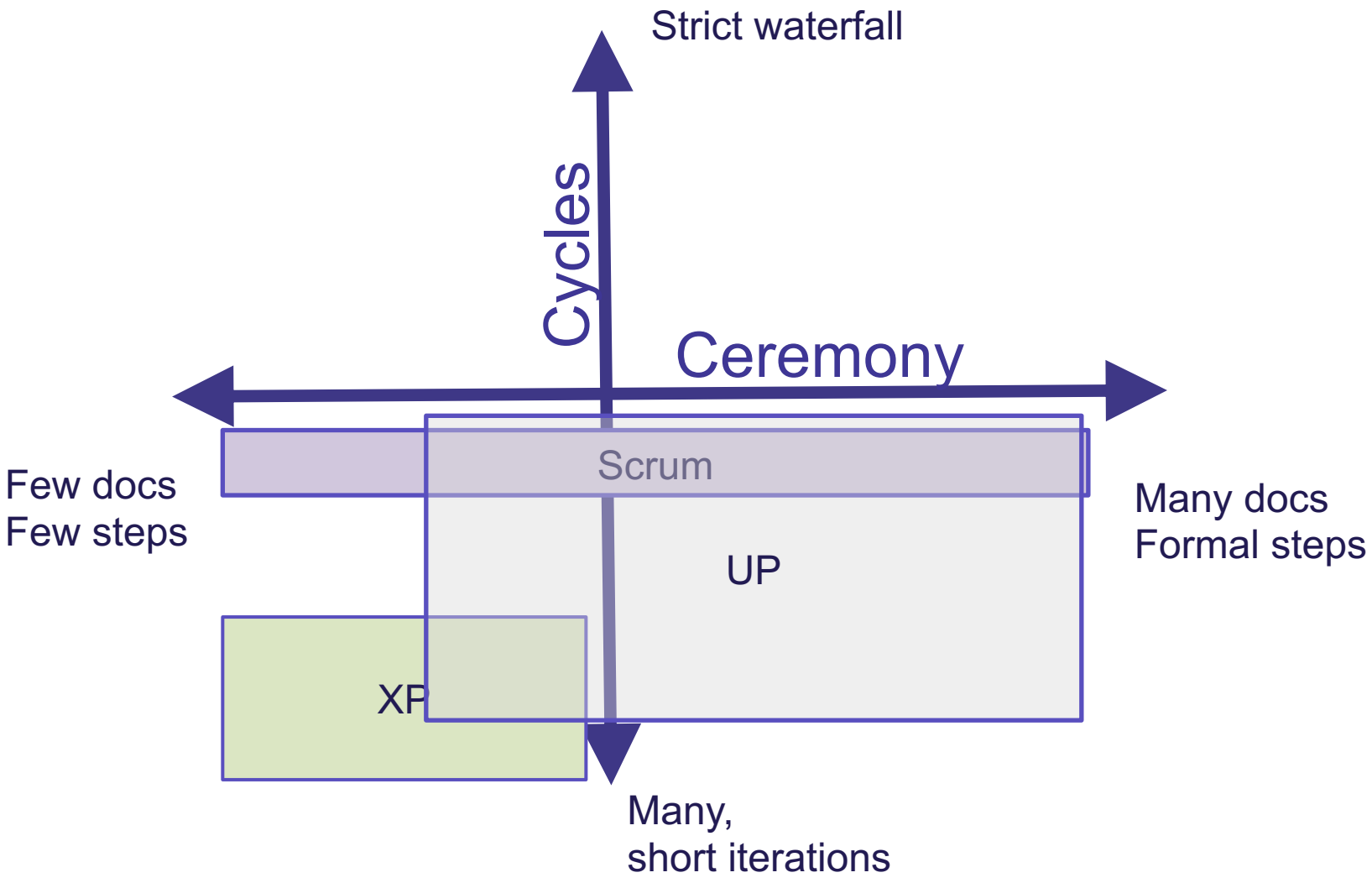
- Incremental development
  - Specification, development and validation are interleaved. May be plan-driven or agile or a mix.



- Integration and configuration
  - The system is assembled from existing configurable components. May be plan-driven or agile.

In practice, most large systems are developed using a process that incorporates elements from all of these models.

Method overview



BALANCING PLAN-DRIVEN AND AGILE

Brooks on silver bullets

- What is a silver bullet?
- Legend says that a werewolf can only be shot with a silver bullet through the heart

- There are no silver bullets in software engineering

Boehm & Turner's Observations

1. Neither agile nor plan-driven methods provide a silver bullet.
2. Agile and plan-driven methods have home grounds where one clearly dominates the other.
3. Future trends are toward application developments that need both agility and discipline.
4. Some balanced methods are emerging.
5. It is better to build your method up than to tailor it down.
6. Methods are important, but potential silver bullets are more likely to be found in areas dealing with people, values, communications, and expectations management.

Home Grounds

	Agile	Plan-driven
APPLICATION		
Primary goals	Rapid value; responding to change	Predictability; stablity; high assurance
Size	Smaller teams and projects	Larger teams and projects
Environment	Turbulent; high change; project-focused	Stable; low change; project/organization-focused
MANAGEMENT		
Customer relations	Dedicated on-site customers, where feasible; focused on prioritized incrememts	As-needed customer interactions; focused on contract provisions; increasingly evolutionary
Planning/control	Internalized plans; qualitative control	Documented plans; quantitative control
Communications	Tacit interpersonal knowledge	Explicit documented knowledge

Home Grounds, contd.

TECHNICAL		
Requirements	Proritized informal stories and test cases; undergoing unforeseeable change	Formalized project; capability; interface; quality; foreseeable evolution requirements
Development	Simple design; short increments; refactoring assumed inexpensive	Architect for parallel development; longer increments; refactoring assumed expensive
Test	Executable test cases define requirements	Documented test plans and procedures
PERSONNEL		
Customers	Dedicated; colocated CRACK (Collaborative, Representative, Authorized, Committed,Knowledgable) performers	CRACKperformers, not always collocated
Developers	> 30% FT Cockburn level 2 and 3 experts; no 1b or -1	50% level 3s early; 10% throughout; 30% 1b's; 0 -1
	Comfort and empowerment	Comfort and empowerment

Cockburn's developer levels

Level	Characteristics
3	Able to revise a method (break its rules) to fit an unprecedented new situation
2	Able to tailor a method to fit a precedented new situation
1A	With training, able to perform discretionary method steps (e.g., sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With experience can become Level 2.
1B	With training, able to perform procedural method steps (e.g. coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience can master some Level 1A skills.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.

Drawing on the three levels of understanding in Aikido (Shu-Ha-Ri), Alistair Cockburn has identified three levels of software method understanding that can help sort out what various levels of people can be expected to do within a given method framework [2]. We have taken the liberty of splitting his Level 1 to address some distinctions between agile and disciplined methods, and adding an additional level to address the problem of method-disrupters.

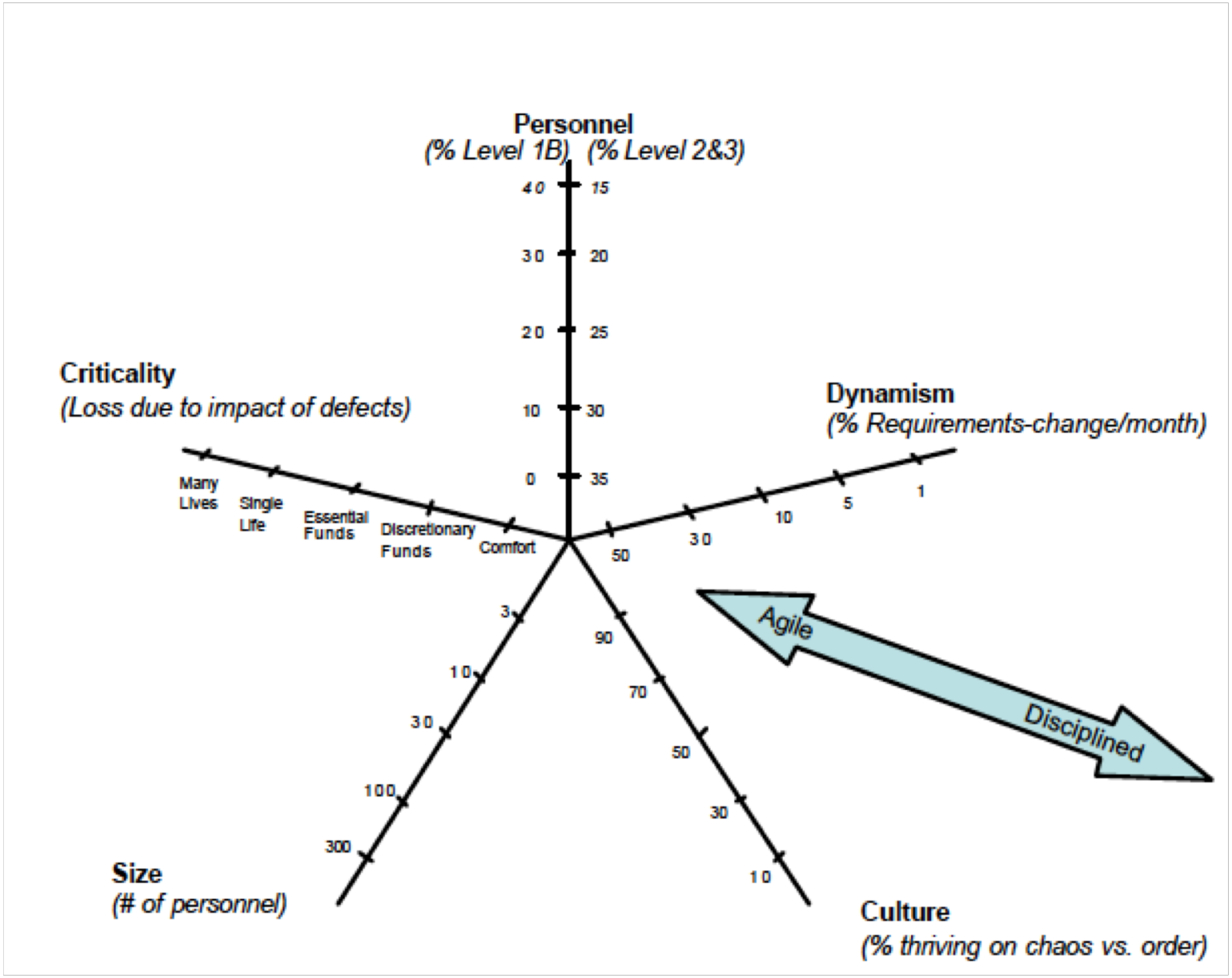
Level -1 people should be rapidly identified and found work to do other than performing on either agile or disciplined teams.

Level 1B people roughly correspond to the "1975-average" developer profile. They can function well in performing straightforward software development in a stable situation. But they are likely to slow down an agile team trying to cope with rapid change, particularly if they form a majority of the team. They can form a well-performing majority of a stable, well-structured disciplined team.

Level 1A people can function well on agile or disciplined teams if there are enough Level 2 people to guide them. When agilists refer to being able to succeed on agile teams with ratios of 5 Level 1 people per Level 2 person, they are generally referring to Level 1A people.

Level 2 people can function well in managing a small, precedented agile or disciplined project but need the guidance of Level 3 people on a large or unprecedented project. Some Level 2s have the capability to become Level 3s with experience. Some do not.

Dimensions


Personnel
(% Level 1B) (% Level 2&3)
40
30
20
10
0
15
20
25
30
35
Criticality
(Loss due to impact of defects)
Many Lives
Single Life
Essential Funds
Discretionary Funds
Comfort
Dynamism
(% Requirements-change/month)
50
30
10
5
1
Agile
Disciplined
3
10
30
100
300
Size
(# of personnel)
90
70
50
30
10
Culture
(% thriving on chaos vs. order)

Group exercises

Assignment 4: Analyse home grounds

- Analyse using Boehm & Turner's home ground technique. Use the table in the article and be specific and detailed.

Assignment 5-8 on Scrum, XP and RUP

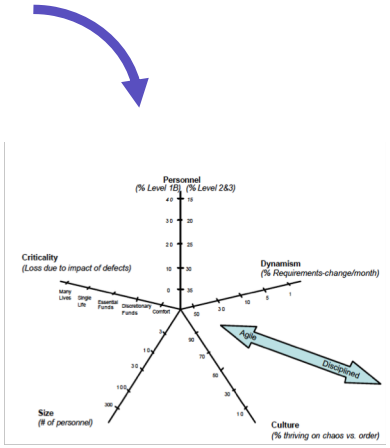
- Timebox to 30 minutes
- Consider, discuss and argue how you would apply each of them to your own project.

Assignement 9

- Compose process for your project, with aregument for mix of methods

Assignment 4 and 9 are more important than assignment 5-8.

Characteristics	Agile	Plan-driven
Application		
Primary Goals	Rapid value; responding to change	Predictability, stability, high assurance
Size	Smaller teams and projects	Larger teams and projects
Environment	Turbulent; high change; project-focused	Stable; low - change; project/organization focused
Management		
Customer Relations	Dedicated on-site customers; focused on prioritized increments	As-needed customer interactions; focused on contract provisions
Planning and Control	Internalized plans; qualitative control	Documented plans, quantitative control
Communications	Tacit interpersonal knowledge	Explicit documented knowledge
Technical		
Requirements	Prioritized informal stories and test cases; undergoing unforseeable change	Formalized project, capability, interface, quality, forseeable evolution requirements
Development	Simple design; short increments; refactoring assumed inexpensive	Extensive design; longer increments; refactoring assumed expensive
Test	Executable test cases define requirements, testing	Documented test plans and procedures
Personnel		
Customers	Dedicated, collocated CRACK* performers	CRACK* performers, not always collocated
Developers	At least 30 percent full-time Cockburn Level 2 and 3 experts; no Level 1B or -1 personnel**	50 percent Cockburn Level 3s early; 10 percent throughout; 30 percent Level 1Bs workable; no Level -1s**
Culture	Comfort and empowerment via many degrees of freedom (thriving on chaos)	Comfort and empowerment via framework of policies and procedures (thriving on order)
* Collaborative, Representative, Authorized, Committed, Knowledgable ** See Table 2. These numbers will particularly vary with the complexity of the application		



Home ground