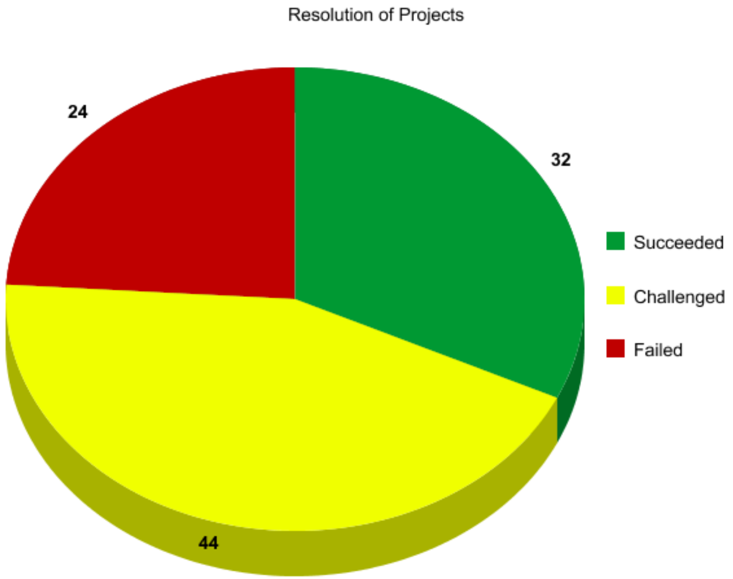
Software Development Methodologies

Lecture 5 - Development Processes

The Standish "Chaos" Report

Reports on statistics about IT projects (data for 2009)

- 32% of all projects succeeded (delivered on time, on budget, with required features and functions)
- 44% are challenged (late, over budget and/or with less than the required features and functions)
- 24% have failed (cancelled prior to completion or delivered and never used)



Among the suspected causes:
poor estimates and poor planning

http://blog.standishgroup.com/

*

Software Development Processes

He who fails to plan,
plans to fail
(Proverb)

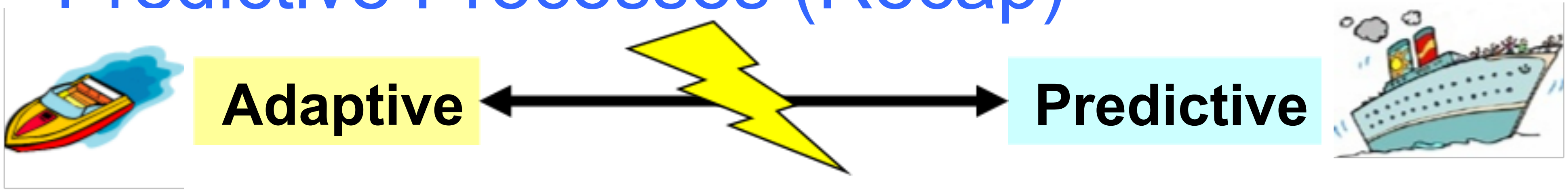
Software Development Process (Recap)

Generic plan for a software project

1. What has to be done? (-> tasks/activities/steps)
2. Why do a task? (-> outcomes, produced artifacts)
3. When should it be done? (-> schedule)
4. Who does it? (-> people, roles, responsibilities)
5. How should it be done? (-> methods, standards, tools)

- Many different processes exist
- No single process suitable for every project (no “one size fits all”)
- Using a process can improve the quality of the product

Adaptive vs. Predictive Processes (Recap)



Adaptive

Predictive

- Lightweight, ‘agile’
- Control by feedback
- Many short iterations (weeks)
- Small scale (<10 developers)
- Face-to-face communication
- Code- & people-centric
- Egalitarian

- Problems:
  - ○ Unpredictable
  - ○ Possible lack of discipline
  - ○ Often underregulated: need to add more rules & practices

- E.g. XP, Scrum, Kanban

- Heavyweight, ‘traditional’
- Control by planning
- Few long iterations (months)
- Large scale (>30 developers)
- Written documents
- Rule-centric
- Authoritarian

- Problems:
  - ○ Inflexibility
  - ○ Bureaucratic overheads
  - ○ Often overregulated: need to select only some of the rules & practices

- E.g. waterfall, RUP

Factors Influencing Software Methodologies

1. Quality requirements
2. Complexity & size of requirements
3. Team size
4. Experience of team members
5. Organizational maturity
6. Technology

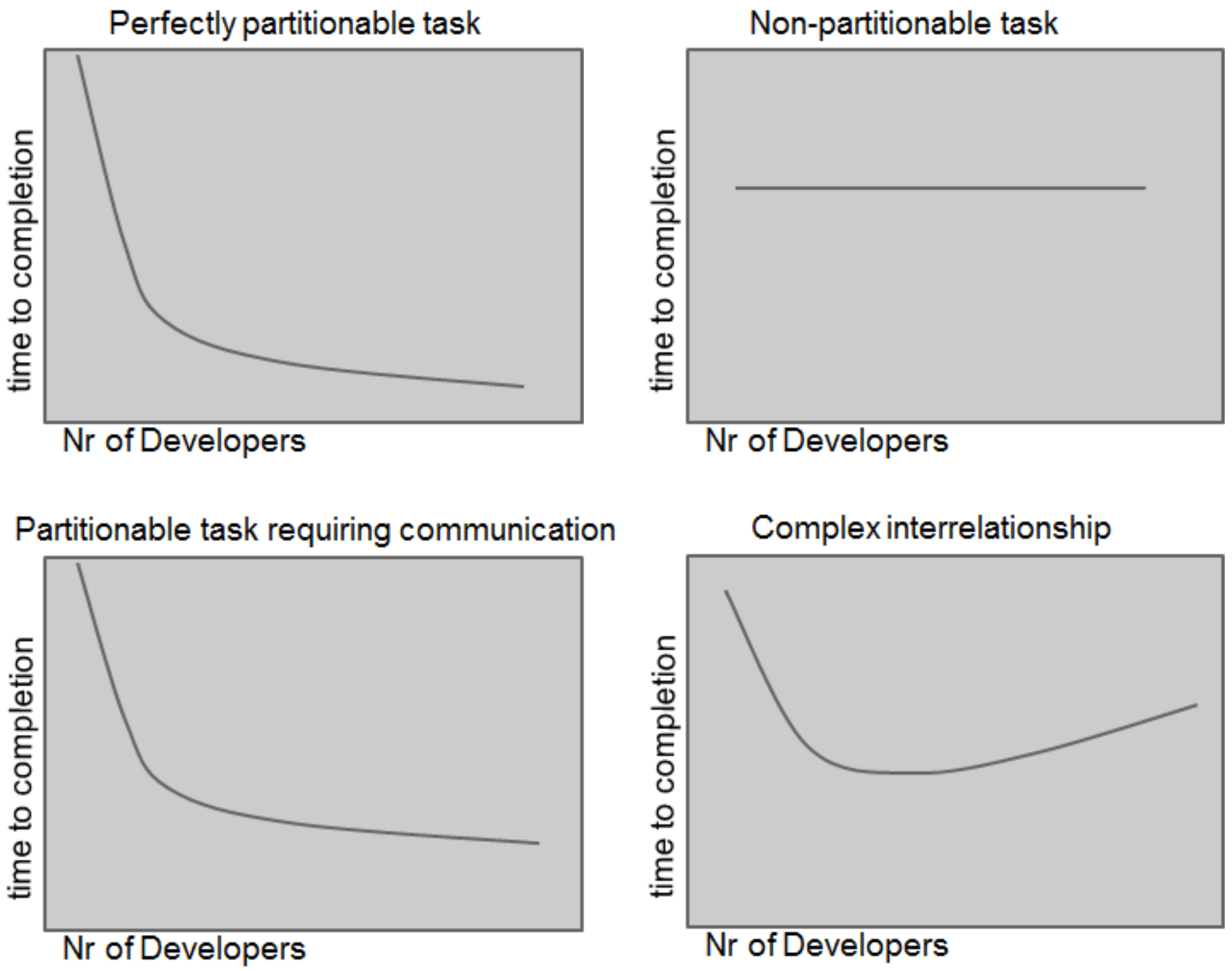
Difficulties in Choosing a Development Process

- Many process models discourage mix & match
  - Complete compliance may be required for certification (e.g. ISO 9000)
  - Claim that only complete compliance reaps benefit, because method elements are mutually dependent

- Commercial interests: Some process models are products or aligned with products & services (e.g. training, certification, tools)

- Bias: publications are not always objective

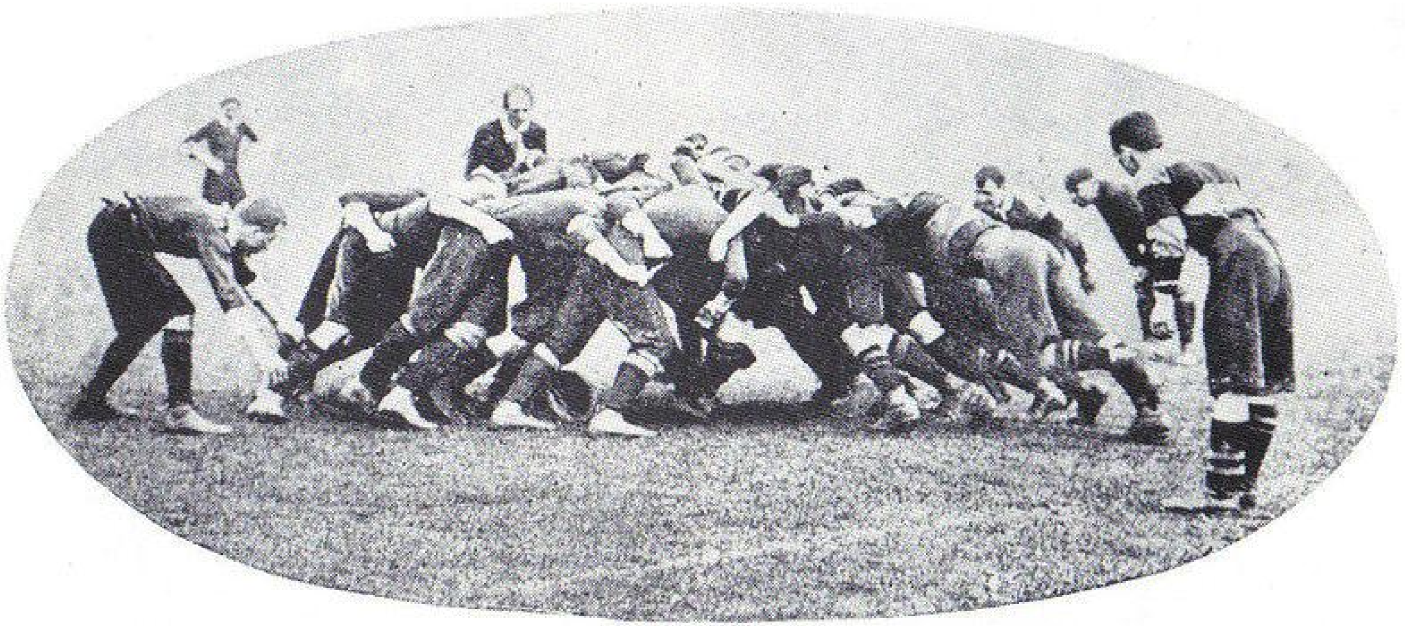
Frederick Brooks: The Mythical Man-Month (1975)



Just adding manpower to a project may not make it faster.

Communication between developers takes time.

Scrum (Recap)



Scrum Theory

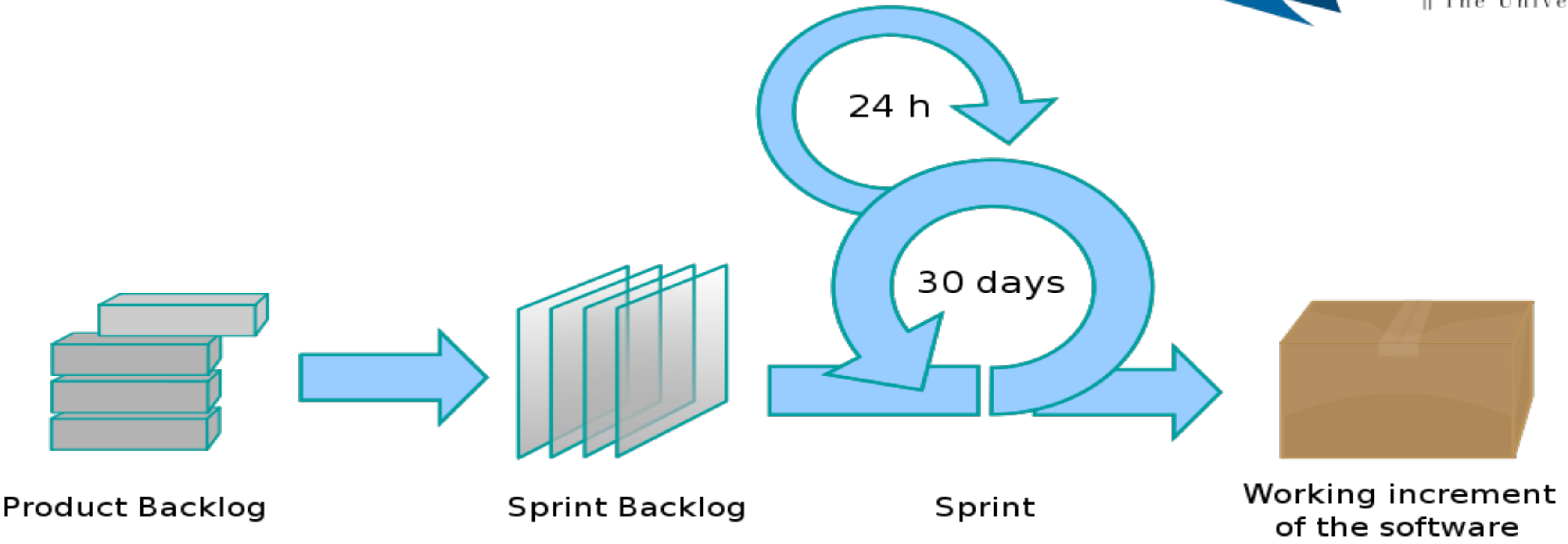
Based on empirical process control theory

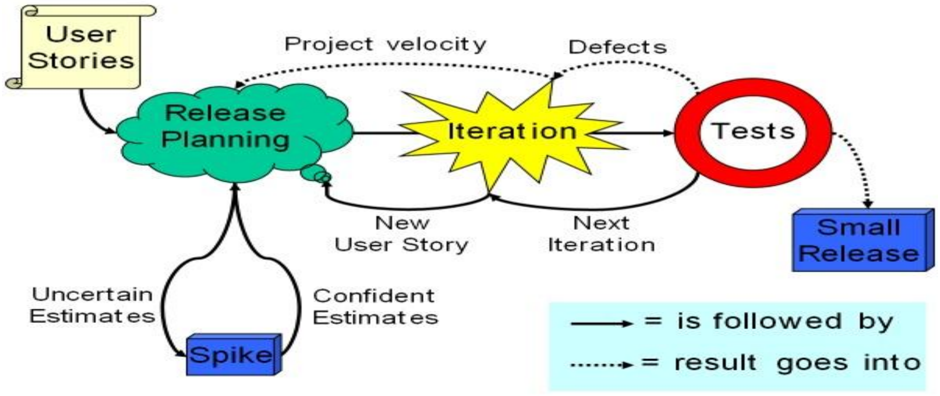
1. **Transparency**: important aspects visible to those responsible
   E.g. common language (definition of "done": tested, reviewed, documented?), availability of information
2. **Inspection**: frequently inspect artifacts and progress
   Detect unwanted "variances" (but don't interfere with work)
3. **Adaptation**: make necessary product & process adjustments ASAP (to avoid waste of work & extra cost)

Scrum provides opportunities for inspection and adaptation:

- Sprint Planning Meeting
- Daily Scrum ("stand-up meeting")
- Sprint Review
- Sprint Retrospective

Scrum Overview





Scrum vs XP:

- Overall process very similar
- Scrum more lightweight: fewer rules & practices
- In particular: no specific engineering practices
- More flexible, but still need to add specific practices for projects (mix & match)

Image by Lakeworks

Scrum Team

- **Self-organizing**: team chooses how to do the work
- **Cross-functional**: all competencies in the team, no outsiders
- **Egalitarian**: no special titles or sub-teams

Product Owner (→Requirements, "What?")

- Responsible for maximizing the value of the product
- Manages the Product Backlog (defining and prioritizing items)
- Decisions must be respected

Scrum Master (→Process, "How?")

- Responsible for ensuring Scrum is understood and enacted
- Facilitates process & communications, coaching developers
  E.g. controls interactions with outsiders (distraction or useful?), troubleshooter

Scrum Artifacts

Product Backlog

- Living document of of **requirements**: ordered list of features/enhancements that are likely needed in product
  - Item attributes: description, order, workload estimate, ...
  - Total work remaining to reach a goal can be summed
  - Product Owner responsible for content & availability
- How do you prioritize? more value, less risk, achievable

Sprint Backlog

- **Set of Product Backlog items**, and plan that defines how the Backlog items are turned into an Increment
- Increment: better product prototype in **usable condition** ("done" as defined by the team)

Scrum Events

Sprint (Iteration, < 1 month)

- Long enough to **get significant work done**
- Short enough to **reduce risk of changing requirements**

Sprint Planning Meeting

- What will be done this Sprint? (**Scoping**)
- How will the work get done? (**Design & Allocation**)

Daily Scrum (daily planning, "stand-up meeting")

- **What has been accomplished**? **What next?**

Sprint Review (product management)

- **Inspect the Increment** and adapt the Product Backlog

Sprint Retrospective (process management)

- **Team inspects itself**: people, relationships, process, tools

Kanban

Take what you need

Introduction to Kanban

- **Scheduling system** created that helps to regulate production
  - Uses signal cards (Kanban="signboard") to signal demand
  - Uses **rate of demand to control rate of production** (to enable lean and just-in-time production)
- Developed at Toyota in late 1940s based on **supermarkets**
  - Supermarket **customers** buy what they need, when they need it
  - Customers only take what they need (future supply is assured)
  - **Supermarkets** only stock what they may sell
- Pass demand **through the supply chain** using Kanban cards
- 1953: Toyota applies Kanban in their main plant

Kanban for Software Development

Summary: visualize workflow, limit WIP, pull work

1. **Visualize the workflow**
   - Write tasks on cards and put on a wall
   - Named columns to illustrate workflow stages
2. **Limit Work In Progress** (WIP):
   set maximum number of tasks for each stage
3. **Measure lead time**: average time to complete one task
   Lead time is tracked, predicted & optimized

TODO	Dev	Test	Release	Done!
I	G	E	C	A
J	H	F	🟩	B
K		🟩		🟩
L				🟩

Start new task only if free slot available

Kanban and Scrum

- Kanban **isn't a software development process**, but can be used as part of one (e.g. combined with Scrum)
- Kanban is **more lightweight than Scrum**
  - No prescribed roles
  - No prescribed meetings or planning activities
  - No timeboxed (i.e. with time constraints) iterations
  - Limits WIP per workflow stage (Scrum limits WIP per iteration)
  - Scrum iterations are less changeable once planned
- What if **Scrum iterations are too long / too inflexible**? Kanban can help to deal with fast-changing requirements and priorities, e.g. in support and maintenance

Kanban Reflections

- Improves **visibility of process** for team & stakeholders
  - Helps to expose bottlenecks & inefficiencies
  - Bottlenecks block process and people can see this
  - Can help to change behavior and improve collaboration
  - Can encourage process improvements
- Allows for **more flexibility**:
  Easier to react to changing requirements
- But...
  - May encourage team to be **too reactive**
    (lack of long-term vision)
  - Encourages **linear workflow** (waterfall-style)
  - Does not account for **task dependencies**
    (in its simple form)
  - Does not encourage overall consistency of **design**
    (but may be augmented with other methods)

Today’s Summary

- **Scrum** is a process framework for agile product development (similar to XP without specific practices)
- **Kanban** is a scheduling & process visualization approach
- Methods such as Scrum and Kanban can (and should) be adapted and combined (**mix & match**)

Further Reading:

- Ken Schwaber and Jeff Sutherland. Scrum Guide. http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf
- Henrik Kniberg and Mattias Skarin. Kanban and Scrum. http://www.infoq.com/minibooks/kanban-scrum-minibook

Quiz

1. What is meant by "cross-functional teams"?
2. What does a Scrum Master do?
3. Describe a possible benefit and a possible risk of Kanban.

```
#define _ -F<00||--F-OO--;
int F=00,OO=00;main(){F_OO();printf("%1.3f\n",4.
*-F/OO/OO);}F_OO()
{
            _-_-_-_
       _-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_-_
        _-_-_-_-_-_-_-_
            _-_-_-_
}
```

What does this C code do? (ioccc.org) (Answer: calculates Pi)