SCRUM In Video Games: A Case Study at High Moon

UNIVERSITY OF CAPE TOWN
Department of Computer Science

JonJon Clark

clrjon005@myuct.ac.za

slides and materials taken from prof edwin blake & Melissa Densmore

Based on Clinton Keith and Colleagues

CTO *High Moon Studios (Darkwatch, The Bourne Conspiracy)*

Scrum at High Moon Studios

- High Moon Studios first video game was Darkwatch, the Vampire-Western first-person shooter for the PlayStation2 and the Xbox
- Produced *Robert Ludlum's The Bourne Conspiracy*.
- Employ 130 people at a facility in San Diego, California
- Check out the article on "a day in the life" at gamesfromwithin.com/a-day-in-the-life

4

GOING FOR SCRUM

Going for Scrum

Adapting Scrum

Summary

Lessons

Agile Development Could Save Your Studio

- Crisis at High Moon Studios:
  - First project was slipping;
  - Nothing was working;
  - Customers were wondering where all their money was going.
- Introduced SCRUM
  - Management less of a bottleneck,
    - talented staff could focus,
    - get on track,
    - straighten out the project
  - finish the game within six months of the original date

Why Change

- Kirsten Forbes (producer)
  - introduced Scrum at Radical Entertainment
  - allows understanding the process of game development in a better way
- “At the end of every project, we do a huge post-mortem.
  - The lament that comes up most often is that the designers change their minds too many times during development,
  - Well, they didn’t change their minds. They learned something about what was fun and adapted the game to accommodate that learning. — That’s exactly what [designers] should be doing.
- The problem is not the designers—it’s our process.
  - We lock ourselves into a schedule that we can’t easily get out of.
  - To make a change in a rigid schedule means we have to find all the tiny areas that those changes ripple through.
  - That’s complex and difficult, so it made us averse to change.”

Agile Game Production Planning

- Publishers are too risk-averse and require a detailed plan for the entire project.
- Scrum developers resist as they want to adjust the priorities to the emerging game.
  - Publishers fear that without schedules, an agile team can iterate endlessly and never finish.
- Agile does *not* avoid long-term planning in favour of only planning a few weeks out.
  - Agile does not support highly detailed long term plans up front.
  - Rather they focus on continual planning for the entire range of the project.
  - Agile continually returns to the assumptions of the plan and modifies them based on reality.

Planning in Scrum

- The results of the last Sprint modify goals and priorities of work in the next Sprint.
  - allows the goals of the project to “drift” a bit every few weeks.
- So if you adopt Agile in industry
  - How can this “drift” exist when the customer demands detailed plans?

Fixed Schedules

- Fixed schedules attempt to predict the priorities and rate of work ahead of time.
- Agile says such predictions aren't accurate and need to revise plans during the project.
  - Planning things up front is not enough.
  - The drift inherent with Scrum can be incompatible with these fixed schedules.

Customers derive security from schedules

- Want a commitment to deliver a product they can use within schedule and budget.
  - Not going to abandon even though they are aware of the problems with fixed schedules.
    - They have seen projects miss those schedules and budgets many times.
    - They need something proven to work better.

Adapting Scrum?

- Challenge for the team using Scrum is to adjust the practice to meet the needs of their customers while preserving the benefits.

Noel Llopis and other engineers at High Moon Studios work in a pair programming environment

Staged Adoption of Scrum

- Many game developers applying Scrum adopt a blend of Waterfall and Scrum.
  - adopt Scrum practices like Sprints, Daily Scrums and Burndown Charts
  - maintain detailed plan documents such as schedules and design documents for long term planning purposes.
  - Sprint practices easy to understand and implement
  - Scrum long term planning practices not easy to grasp.
    - Some Scrum adopters or customers will not trust abandoning long-term schedules completely.

Partial Scrum?

- How well can Scrum work with customers who require detailed schedules without variation?
  - Scrum will be less effective!
    - Some Scrum practices, such as daily stand-up meetings, and regular delivery of a working game, will still help.
  - One benefit gained is customers assume control over the schedule and budget by manipulating the scope.
  - If the customer decides the features built are sufficient for the money spent or if delivery date demands, then the team can move to building the production assets and complete the game.
    - benefit is features of less importance are abandoned
    - or having to cut key features that might be 90% complete.

Rolling Schedules

- Most customers acknowledge problems with trying to create detailed schedules.
- many development contracts have a rolling level of detail for future milestones.
  - An example is a contract specifying a high level of detail for the next milestone, the level of detail dropping for milestones further out.
  - As the project progresses, upcoming milestone details are fleshed out with more detail of what the team hopes to deliver.
- Look like Sprint cycles in a Scrum project.
  - offer little room for the backlog to change within a milestone/release cycle.

ADAPTING SCRUM

Going for Scrum

Adapting Scrum

Summary

Lessons

Adapting Scrum Schedules

- Need to use Scrum for products where there is not a “shippable version” every Sprint.
- This often leads to the demand for long, detailed schedules.
- Lets look at what is wrong with such demands …

Uncertainty in Schedules

- A problems with long term schedules is they try to plan away uncertainties
- Agile methodologies were created for cycles of product development that have a high level of uncertainty.
- Three categories of uncertainty …

Uncertainty in Schedules: *Uncertainty in what you are building.*

- The specifications of what makes a fun game are impossible to document and schedule.
  - "Finding the fun" takes many iterations and much experimentation.

Uncertainty in Schedules: *Uncertainty in technology.*

- Developers face a rapidly changing technology base with products that require major innovation.
  - Games consoles, smartphones
- Many products have been delayed or cancelled because of optimistic predictions about what new technology could achieve or how long it would take to be implement.

Uncertainty in Schedules: *Uncertainty with the developers.*

- Given the same design document and schedule, two different teams will have entirely different results and levels of success.
  - Teams gel differently and talent is highly variable.

Pre-production and Production

- Projects often divided into pre-production and production
  - Can adopt
    - an iterative approach to pre-production
    - followed by a detailed production schedule.
- For certainty in production schedules need to address all three areas of uncertainty during pre-production.
  - Once you do this then pre-production is complete.
  - Pre-production gives a shippable demo version:
    - a percentage of all scope
    - demonstrates the final user experience
    - demonstrates performance
    - specifies resources required to build remainder of the product

SUMMARY

Clinton Keith's Summary of Scrum given at GDC 2006
"Agile Methodology in Game Development, Year 3"

The Scrum Cast

Product Owner

Director

Marketing

Publisher Producer

Customers

Scrum Master

Anyone. Not a lead role

The Team

Artist

Artist

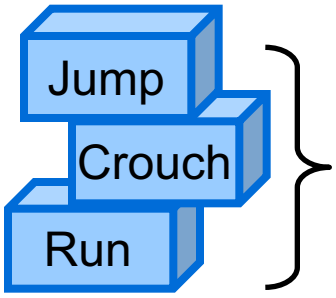
Animator

QA

Designer

Programmer

Product Backlog

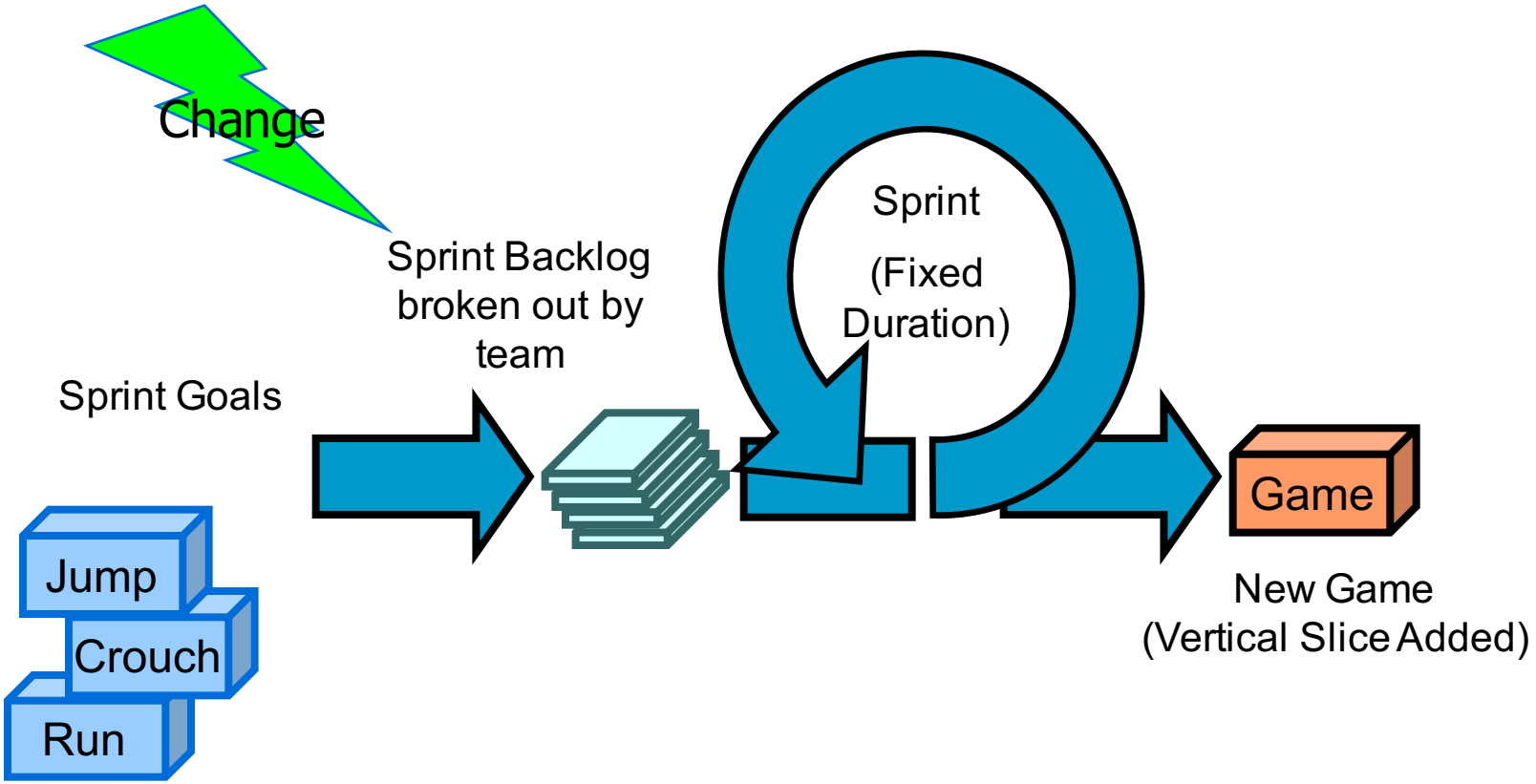


Product Backlog as prioritized by Product Owner

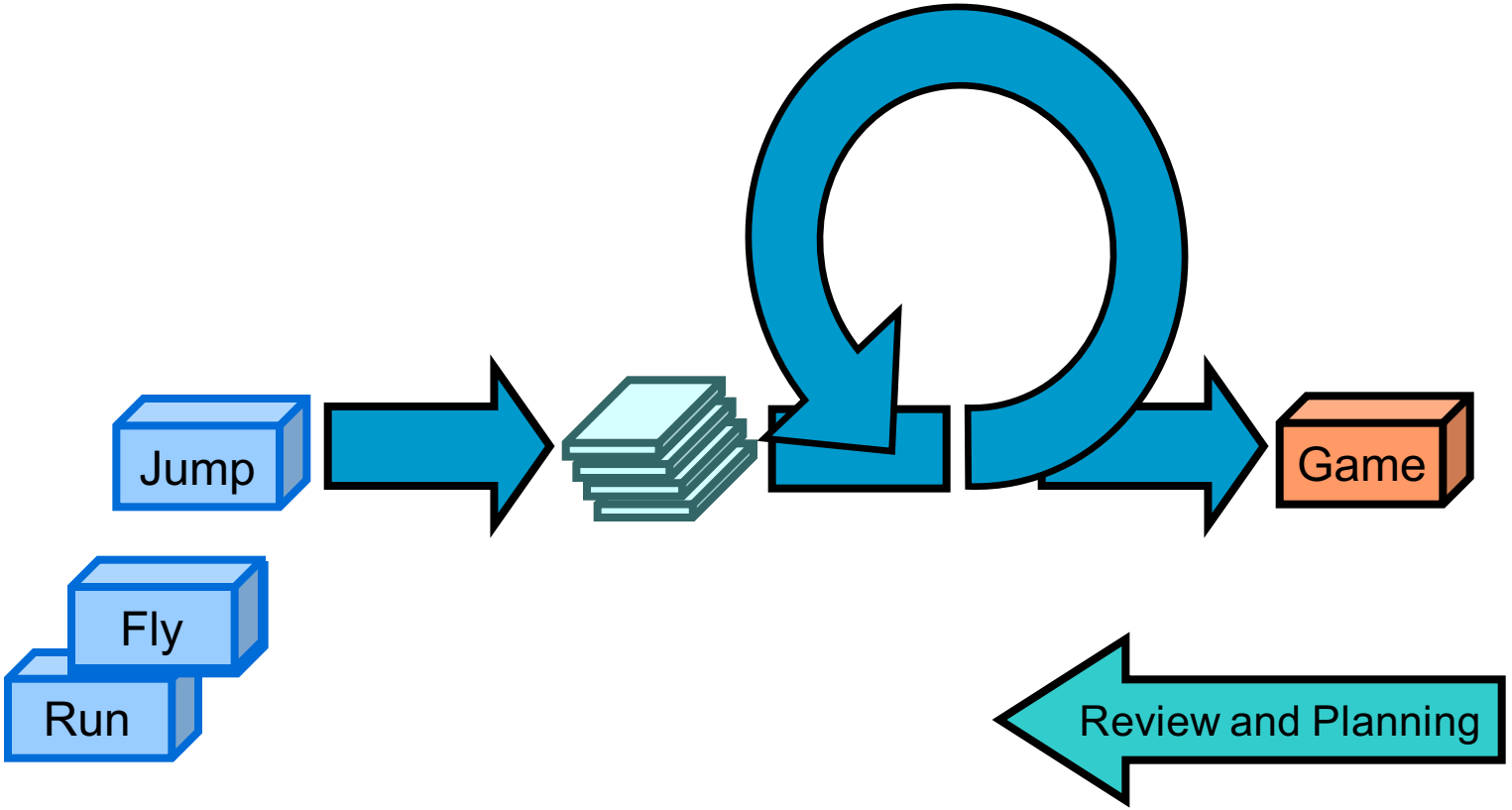
Defined as *User Stories* with conditions of satisfaction

Estimated with relative *User Story Points*
that help track progress

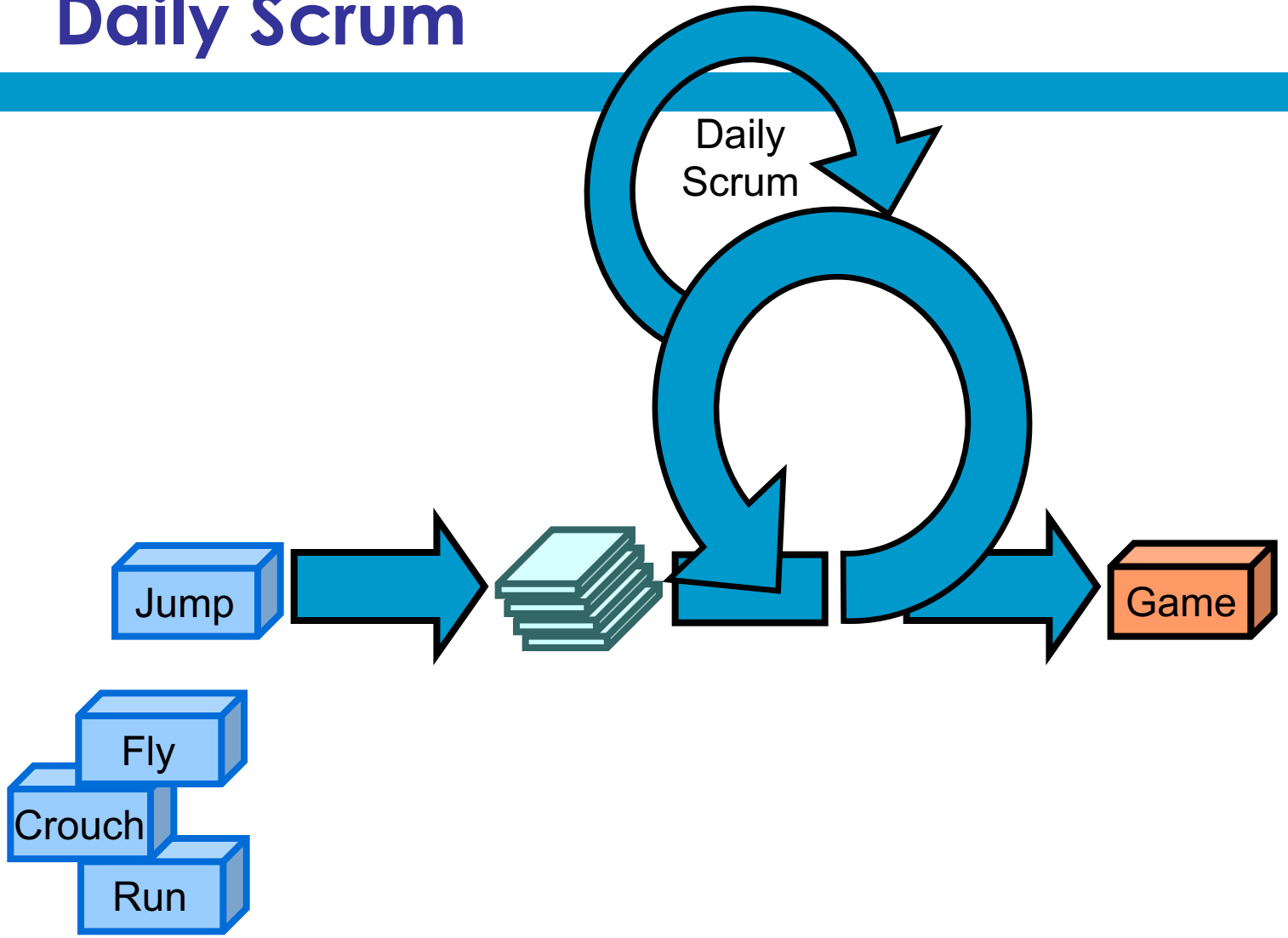
Sprints (Iteration)



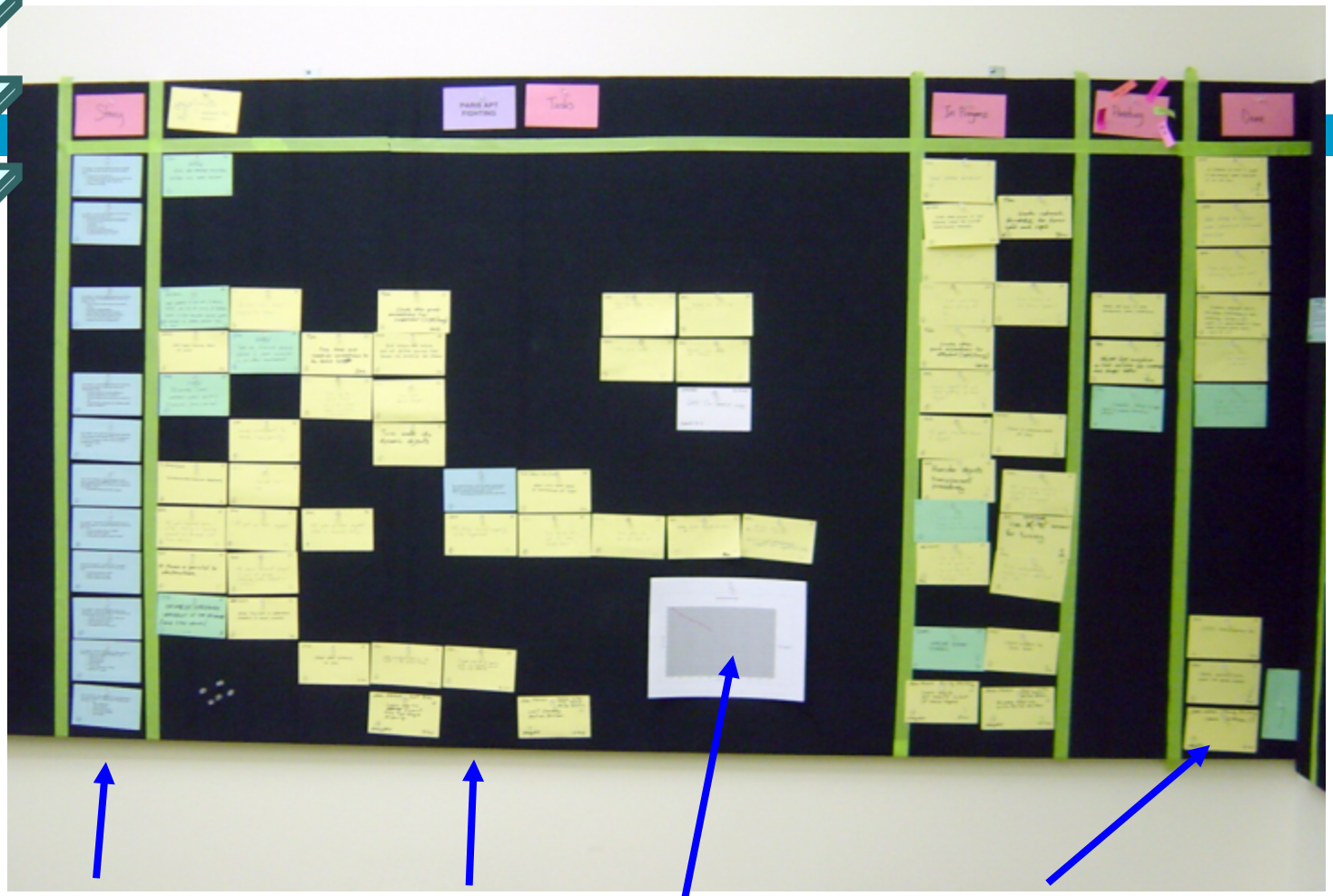
Review and Planning



Daily Scrum



The War Room



User Stories
(Sprint Backlog)

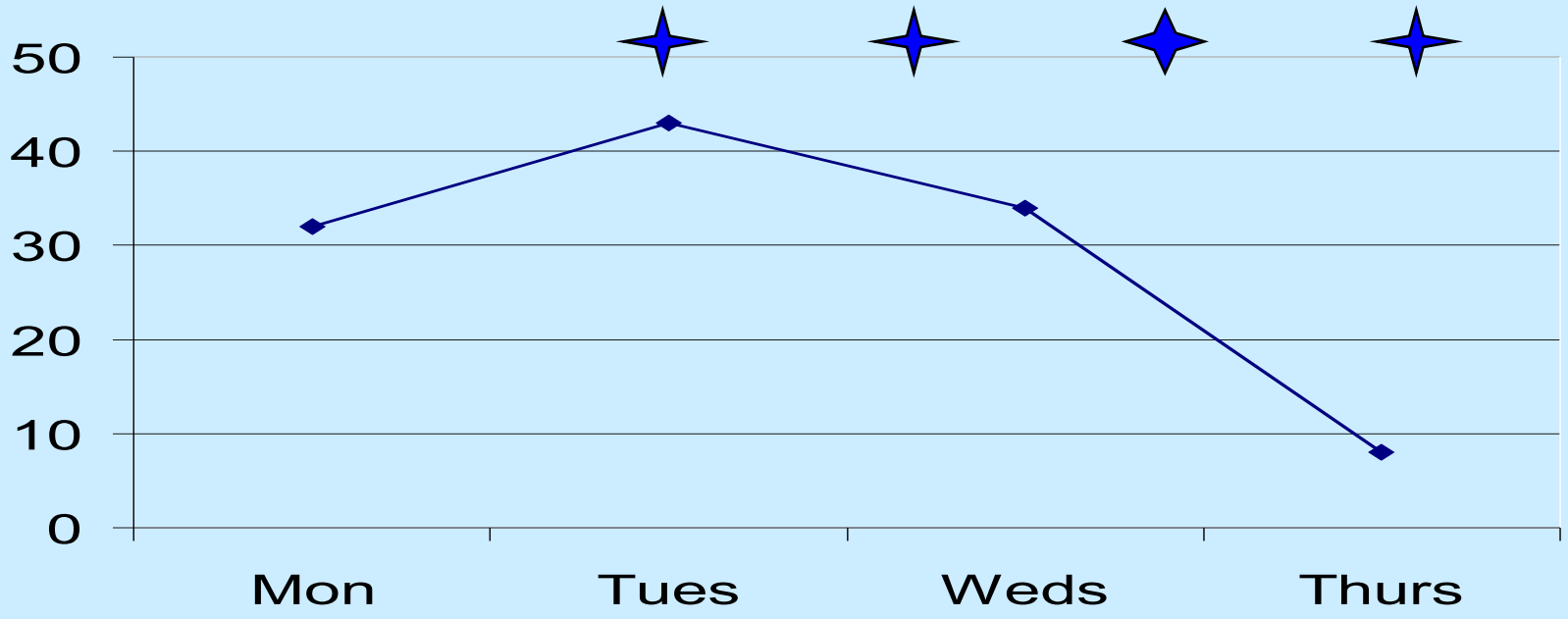
Tasks

Burndown Chart

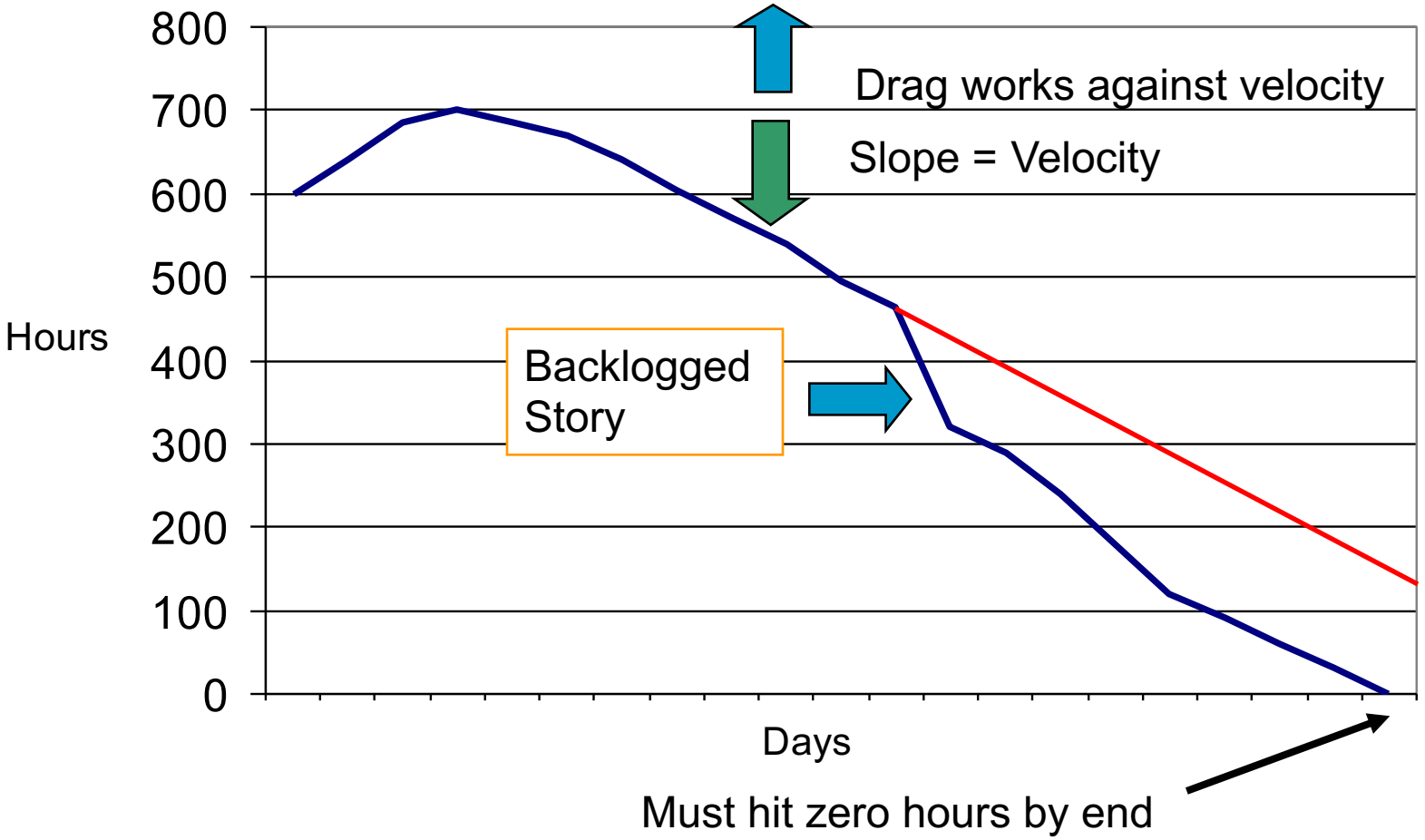
Completed Tasks

Jump User Story

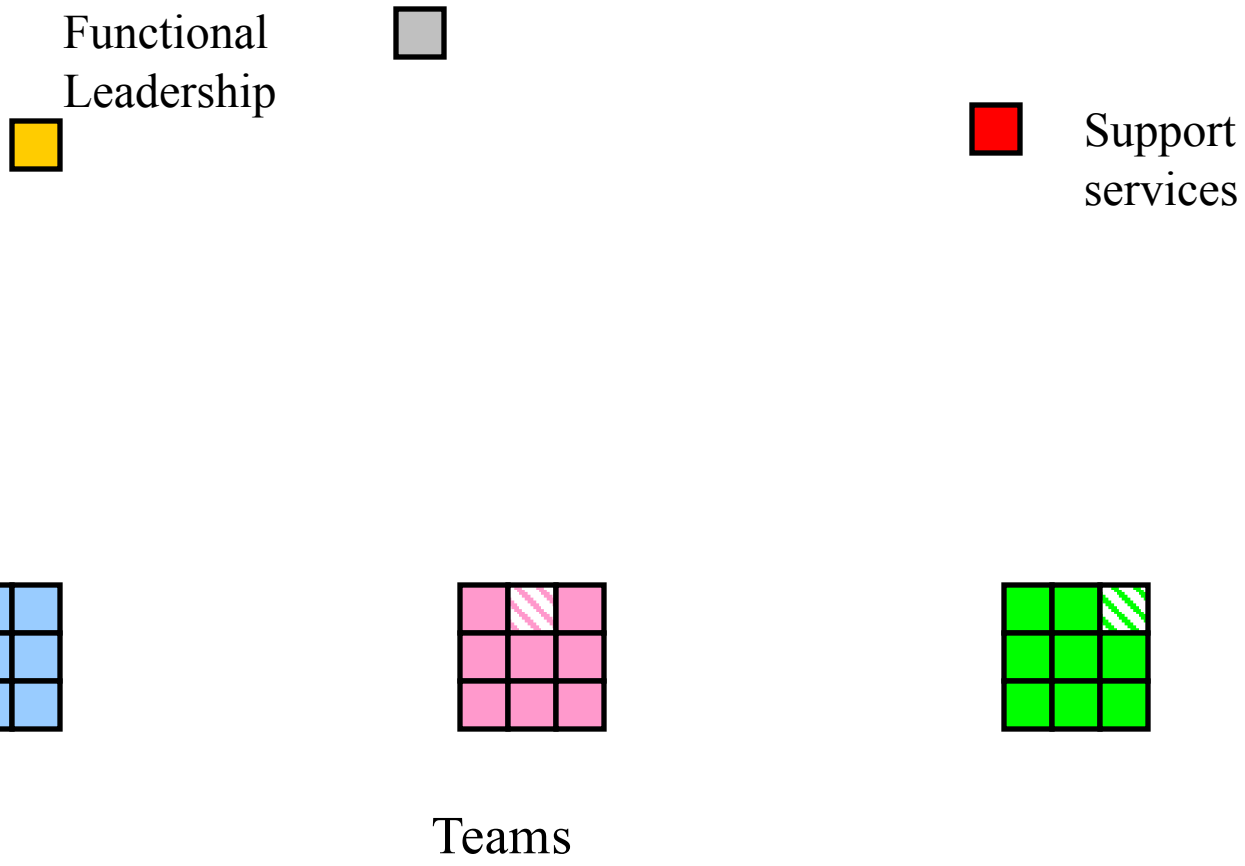
Task	Mon.	Tues.	Wed.	Thurs.
Jump Input Control	8	3	0	0
Jump Tuning	16	16	10	4
Jump Animation	8	16	16	4
Double Jump Option		8	8	0



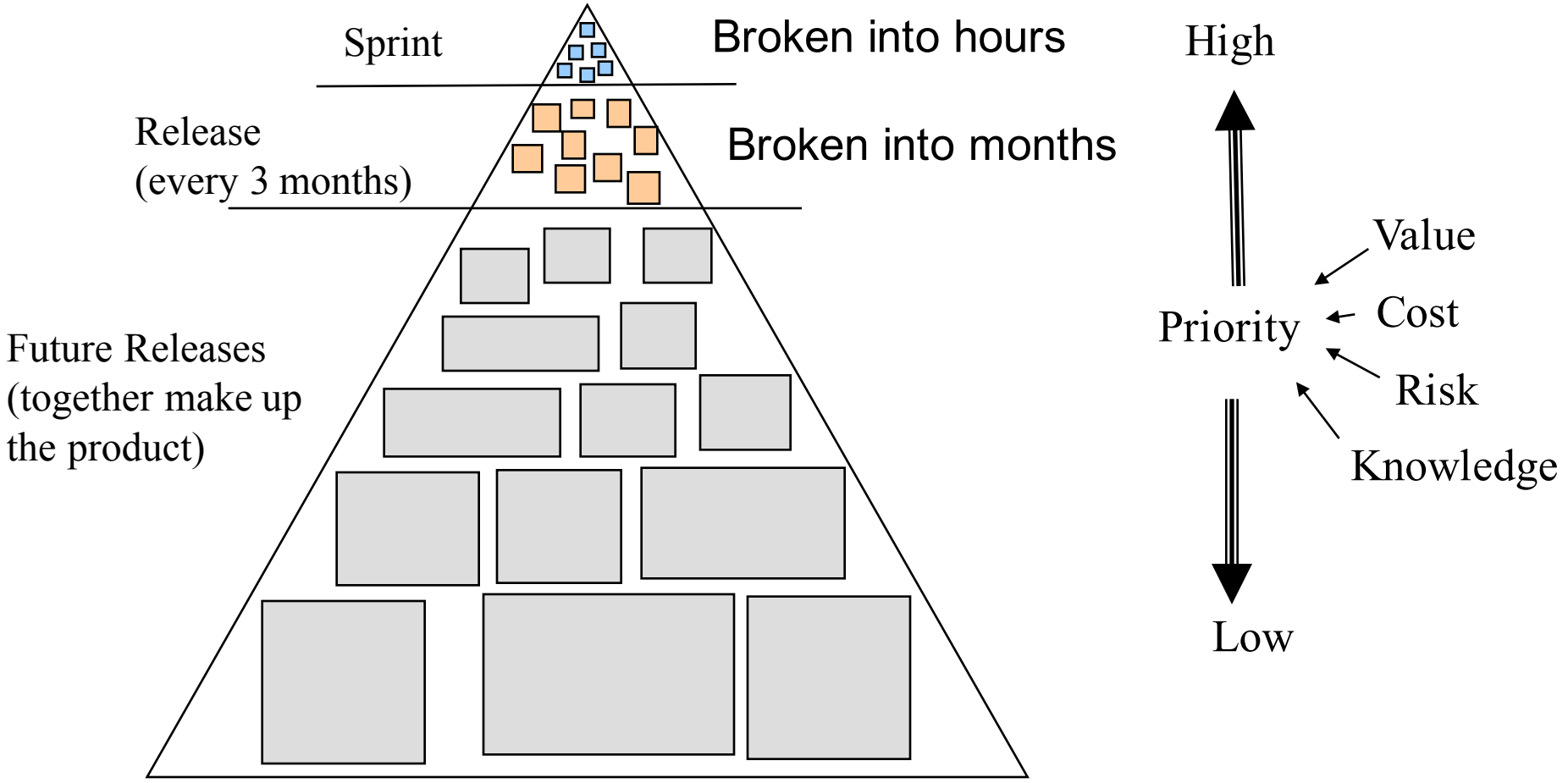
Sprint Backlog Burndown Chart



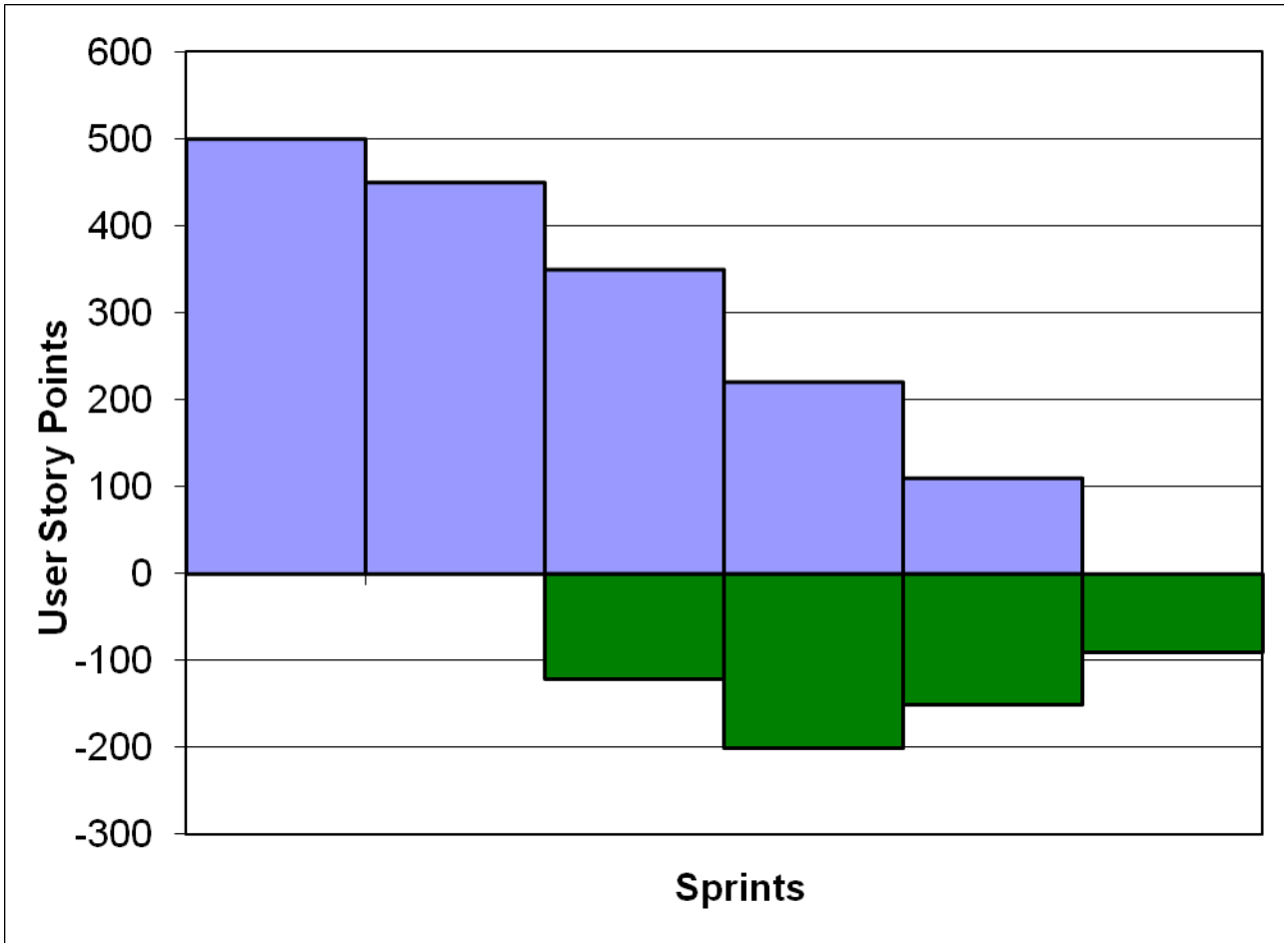
Scaling Scrum — The Scrum of Scrums



Releases — The Product Backlog *Iceberg*

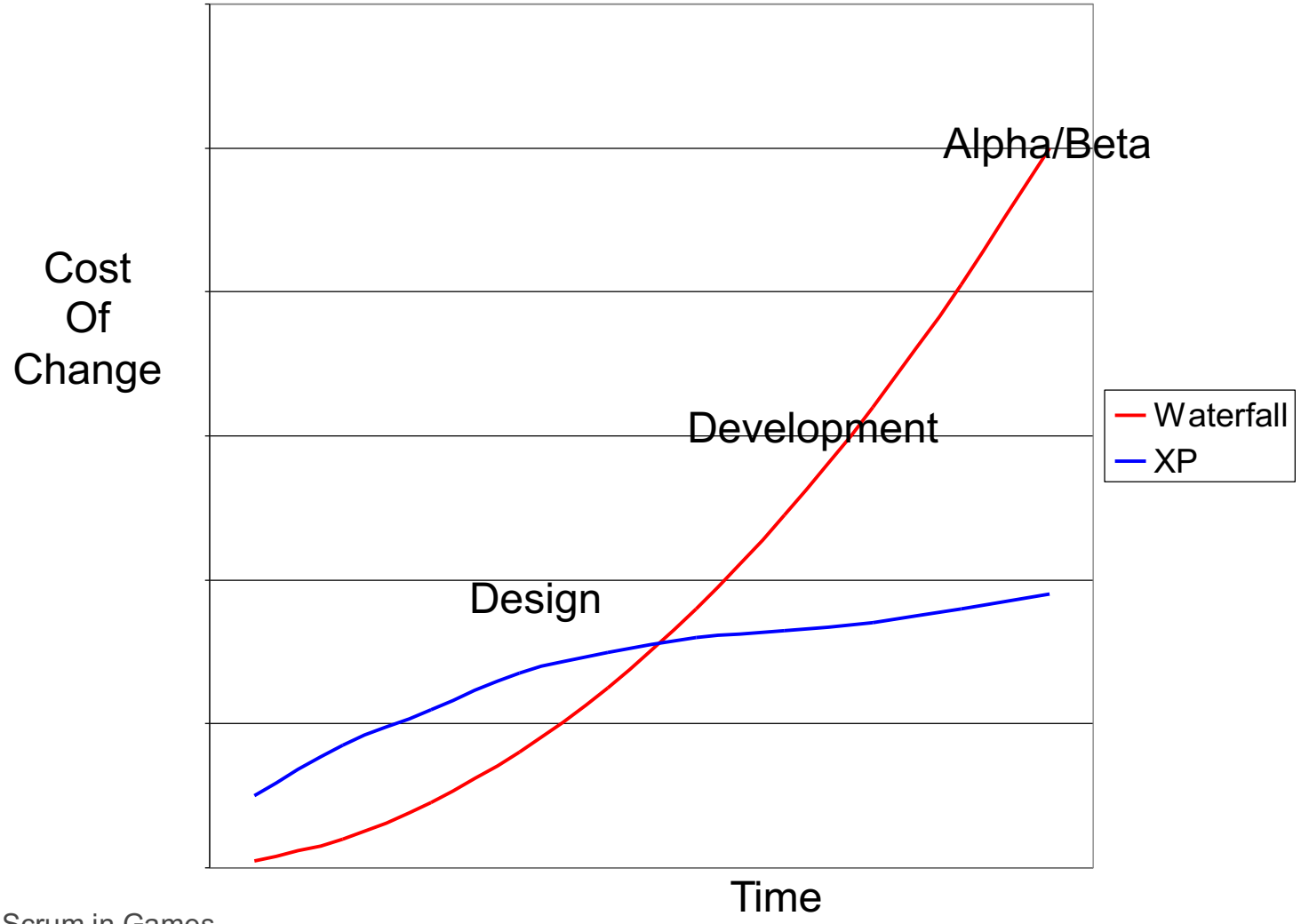


Release Burndown



Progress is shown above the baseline; changes in scope below the baseline

eXtreme Programming (XP)



LESSONS

Lessons

Lessons Learned

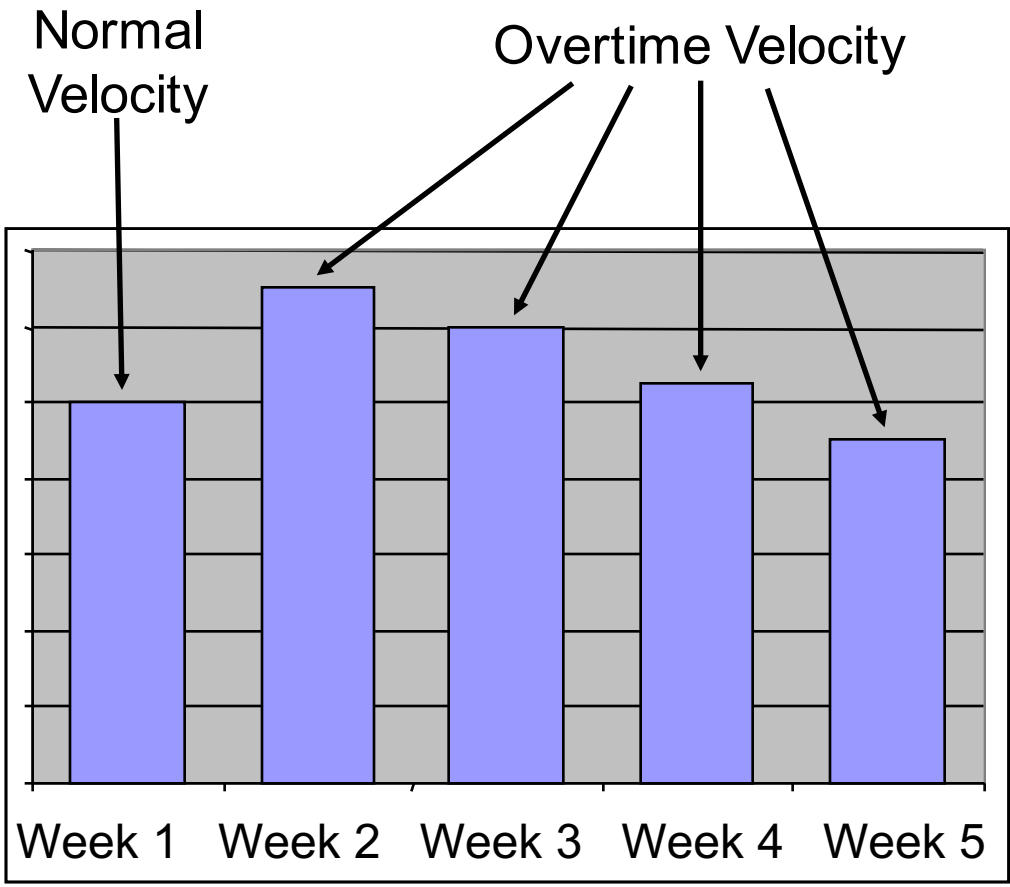
- Start small
  - One Scrum team doing a prototype would be ideal
- Do what the book says from the start
  - Company dysfunctions can creep in otherwise
  - Don’t get too dogmatic about it
  - …but Scrum is meant to be modified (eventually)

Lessons Learned…

- Agile planning is harder to start than thought
  - But XP was easier
- Coaching was critical
  - Onsite and CSM
- Publisher buy-in wasn't difficult
  - Getting them into reviews and planning took adjustment

Lessons Learned…

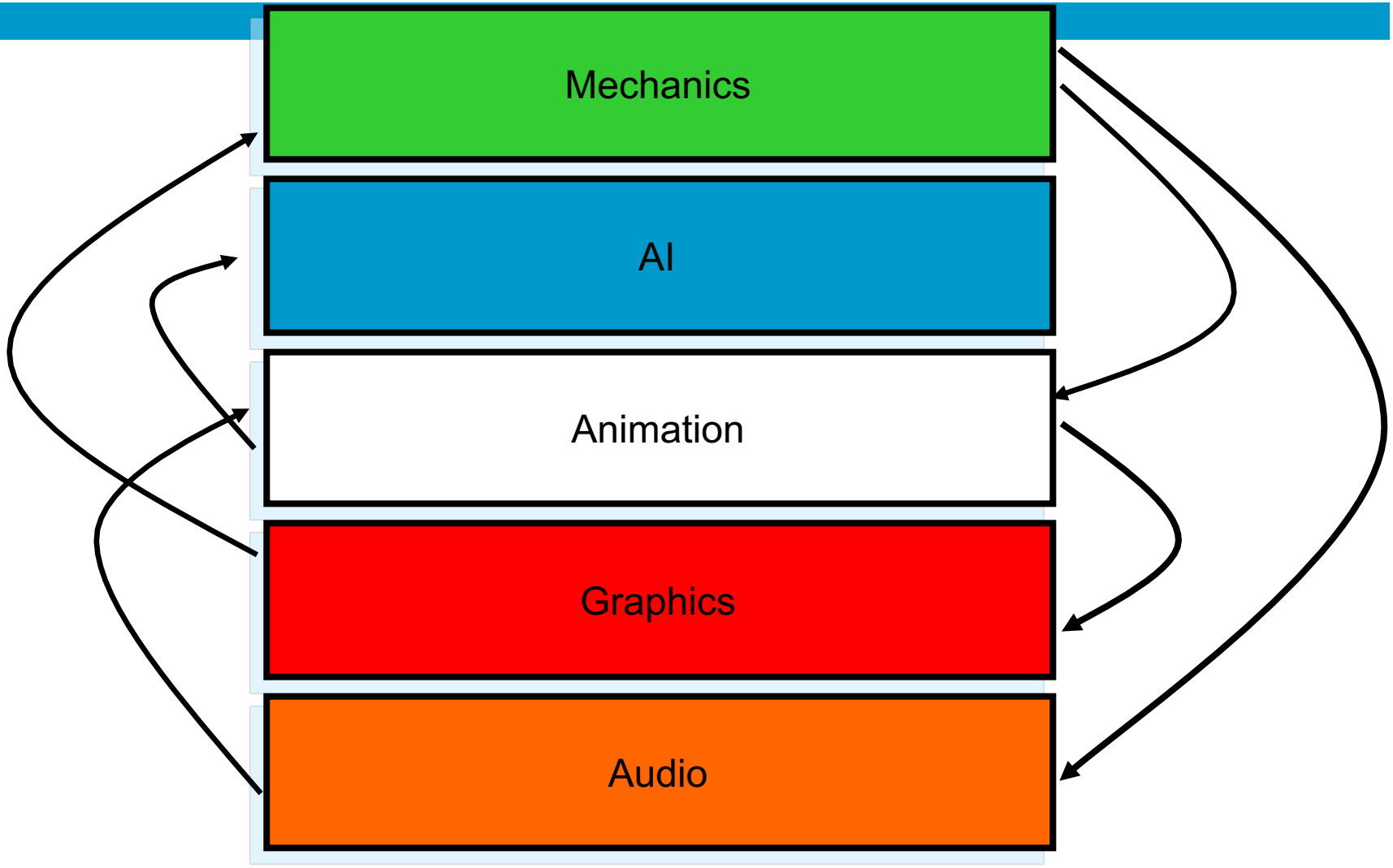
- Overtime value is limited
  - but average intensity is raised
- Old Habits Die Hard
  - Over-design
  - Delayed integration
  - Seating by discipline
  - Command and control
- Testing needs to be pushed



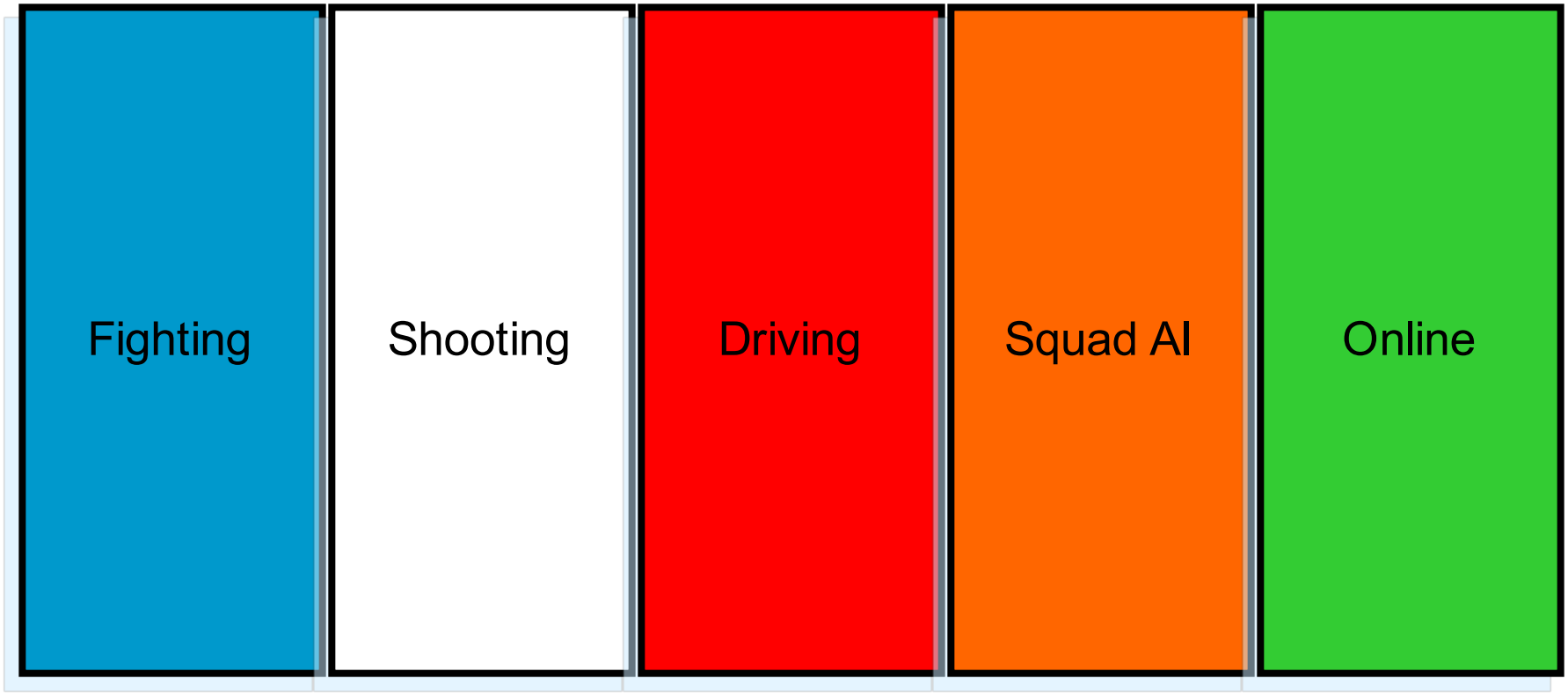
Large teams

- A large project group can lack a sense of ownership divided across many teams
- How does project group break into teams?

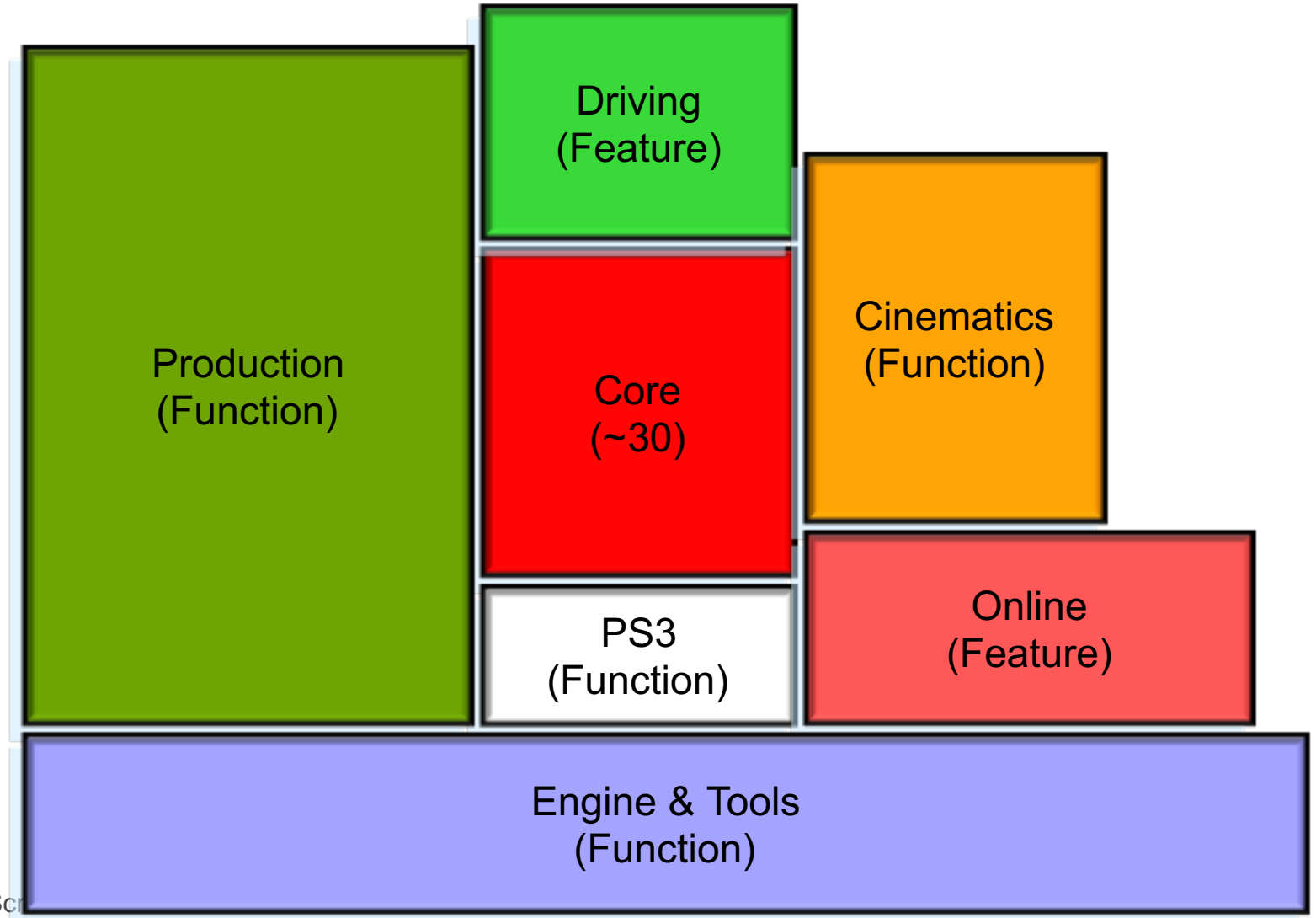
Solution #1: Functional Teams



Solution #2: Feature Teams



Solution #3: Core Team & Dedicated Function & Feature Teams



Pros

- Convenient communication with customers
- Progress tracking (motivating)
- Easier for us to schedule larger tasks!
- Gives name to what we've been doing
- Reduces randomization, increasing efficiency
- Development team's involvement with scheduling
- Daily status emails
- Predictability of QA builds

Cons

- Treadmill effect
- Increased minimum time between QA builds, in practice increased frequency
- Increased minimum latency for non-blocking issues
- Code locking has impact on teams
- Crash report spike after lock is lifted
- No penalty for over/ under- estimating due to punting