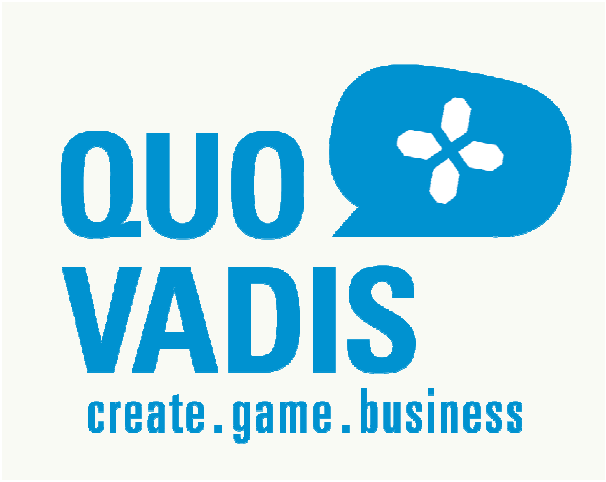


Beyond Agile

Pitfalls and misconceptions when working with SCRUM & Co in video game development

Ralf C. Adam

Disclaimer

This lecture was first held in April 2015 at the Quo Vadis Game Developer Conference in Berlin/GERMANY. The presentation may not include all original materials shown during this speech (pictures, videos etc.) and is solely meant as a summary and handout. All copyrights of presented materials belong to their respective owners.

About Myself

- Ralf C. Adam
- Executive Producer at flaregames
- >20 years in the games industry
- Contributed to >80 games
- Tutor at Games Academy in Berlin & Frankfurt
- Hobbys: Movies, Games, Gadgets, Soccer, Karaoke

About This Talk

- This talk is about agile project management methods and concepts and how to apply them to game development
- I will give examples of what to avoid – as well as some best practices

There‘s no such thing as doing Agile!

What Is Agile?

- Correct Wording:
  - Agile development practices or methods
  - Agile development teams
  - Develop software with agility
- Agile is an adjective not a noun
- It‘s an attribute to something we do…
- …not ***THE*** something

What Is Agile?

- Right now big hype around it
- Everybody talks about „being agile“
- Often feels like buzzword bingo
- Also lots of money involved

  (Trainings, expensive certifications etc.)
- Originally comes from traditional IT
- Question: Is it really 100% adaptable

  for video game development?

What Is Agile?

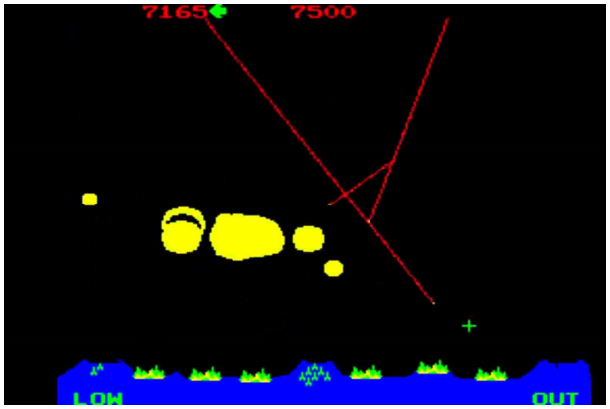
[*agile*]: Relating to or denoting a method of project management, used especially for software development, that is characterized by the division of tasks into short phases of work and frequent reassessment and adaption of plans: *agile methods replace high-level design with frequent redesign*. Contrasted with waterfall (adjective).

SOURCE: oxforddictionairies.com

What Is Agile?

- By the way: It is not a new invention!
- Game developement has always

  been an agile process

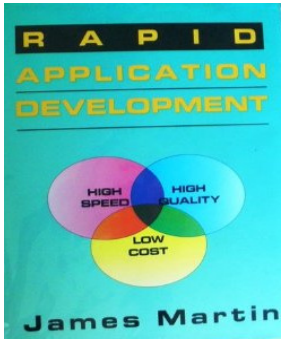


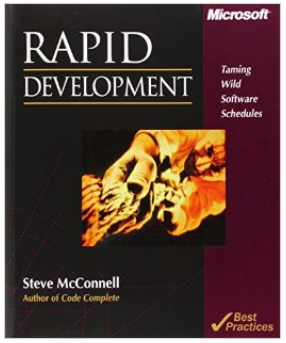
»Pick an idea. Write up a game proposal. Get it OK'd by management. Take a couple of weeks to bring up a playable simple version. Management reviews that and OKs it or axes it. If OK'd, continue with the whole game. Regular reviews by management to make sure still fun. Kill the game if not.«

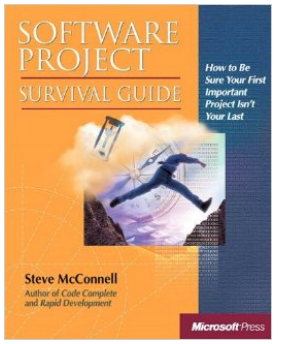
Dave Theurer (1980)
Designer of *Tempest* and
Missile Command

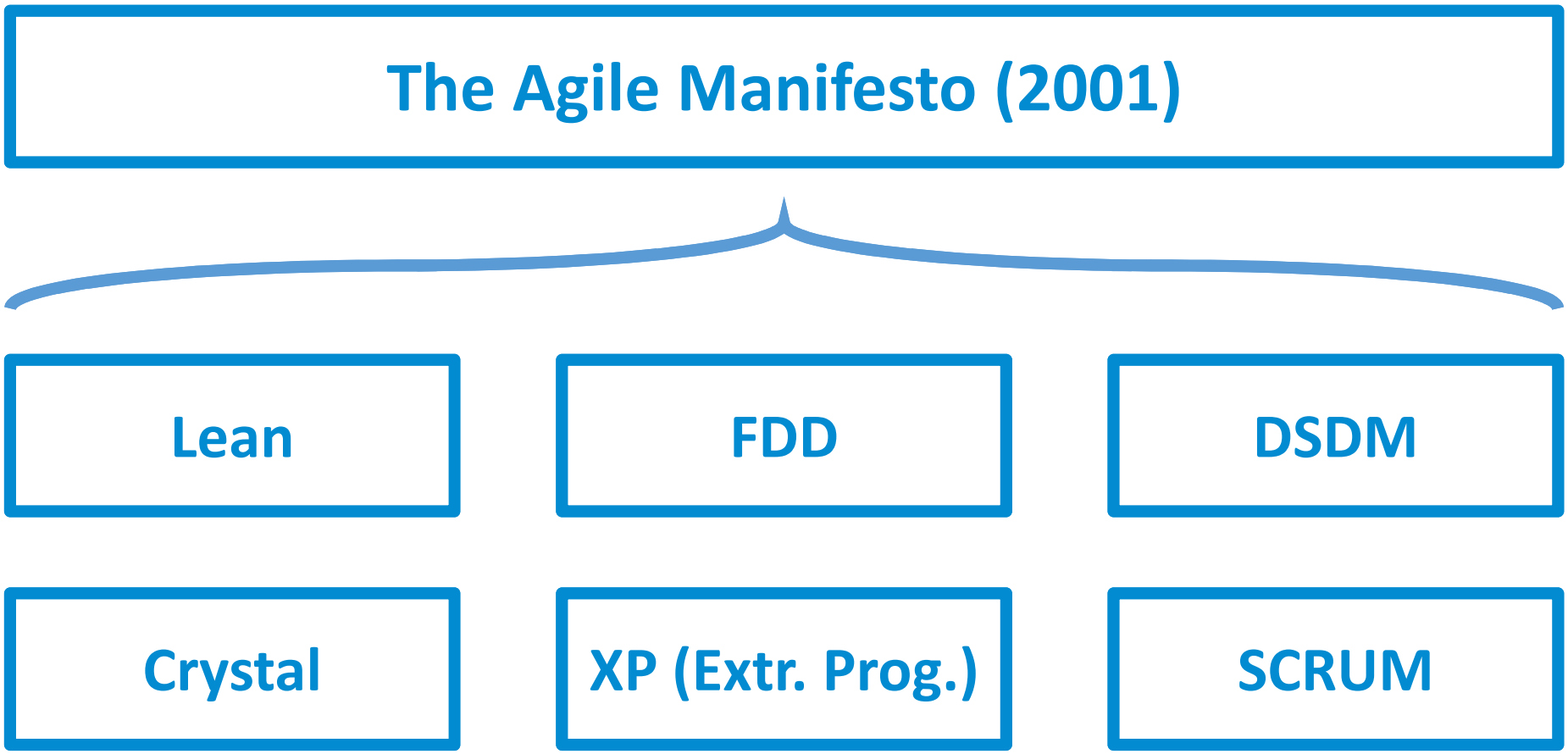
Before Agile There Was RAD

- In 1991 James Martin wrote „Rapid Application Development“ (RAD)
- Princicples described in there:
  - Risk reduction
  - Early prototyping
  - Finding problems earlier in life-cycle (cheaper to fix)
  - Emphasizes necessity of adjusting requirements throughout development
  - Also emphasize a flexible process and planning
- See also: Rapid Development – Steve McConnell (1996)








The Agile Manifesto (2001)
Lean
FDD
DSDM
Crystal
XP (Extr. Prog.)
SCRUM

Lean

- Father = Taiichi Ono (1988), Production Lead Toyota
- Comes from Lean Manufacturing. Set of principles for achieving quality, speed & customer alignment
- No set of fixed Rules or Methods
- It‘s a Philosophy/Company Culture
- Lean Company e.g. can use SCRUM as specific method
- Princicples:
  - Eliminate Waste
  - Improve Quality
  - Reduce Costs
  - Deliver Fast
  - Respect People
  - Constantly improve Skills & Production Processes

- Feature Driven Development
- High Level Method
- Allows for other methods on lower level
- Purely focusses on the Delivery of Features

FDD

- Advancement of RAD
- Rather strict set of Rules
- Acknowledges cooperation with e.g PRINCE2 & PMI
- Focus on Timeboxing and MoSCoW prioritization (Must have, Should have, Could have, Won't have)

DSDM

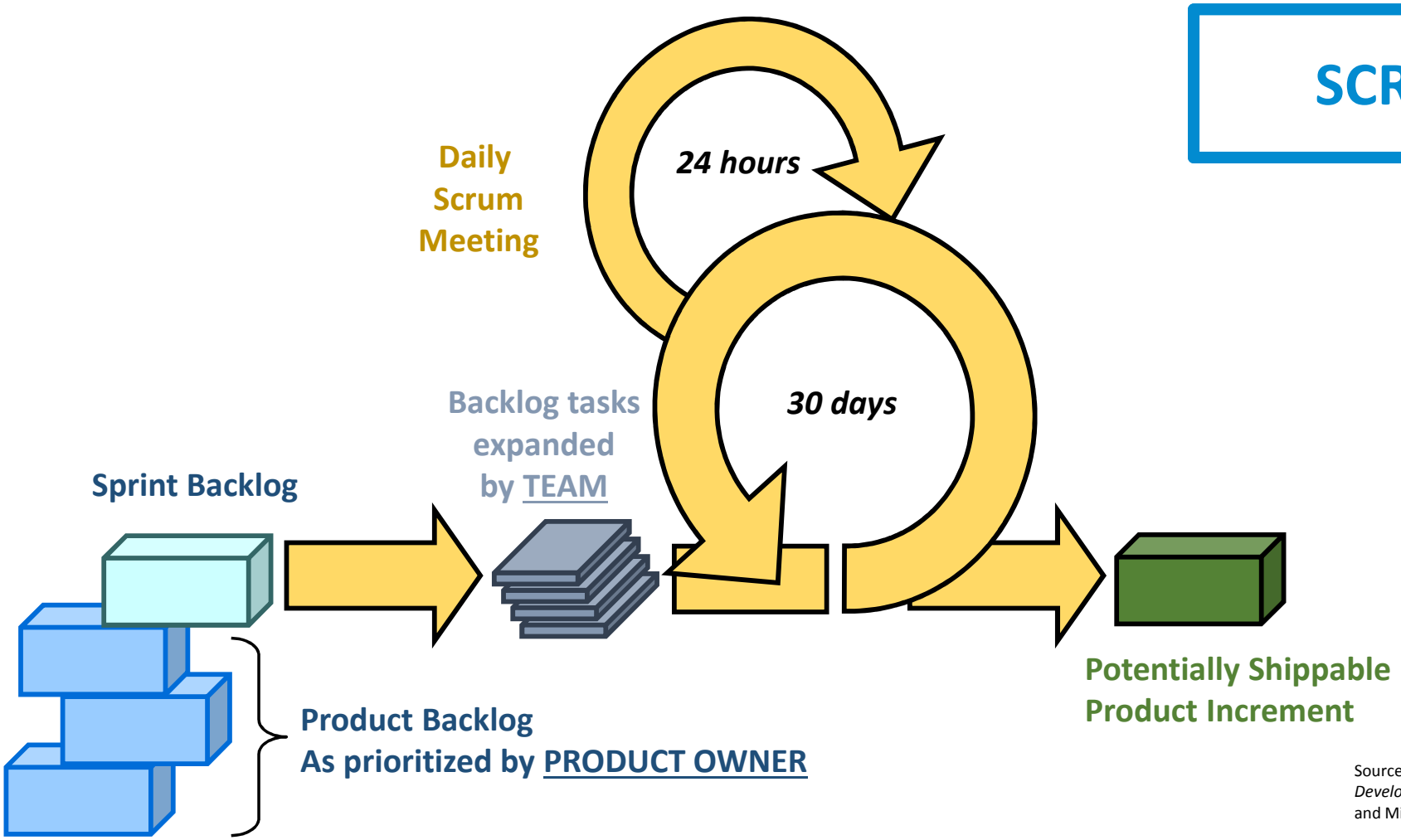
- Developed in 1990'ies by Cockburn
- Focused on people, iteration, community, skills, talents, communications
- More framework than rule set
- Creed: People above process
- Very tolerant

Crystal

XP (Extr. Prog.)

- Focus on Programming
- Emphasizes Team-Work
- Focus on changing customer requirements
- More open to change (than e.g. SCRUM)
- Focus on certain coding practices such as Testing, Feedback, Pair Programming, Simplicity, Test Driven Development etc.

SCRUM



Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle

KANBAN

- Method for Steering Production
- Once Again: Toyota/Taichi Ono
  (Part of the Lean Process)
- Japanese: word Kan means "signal" and "ban“
  means "card"
- Therefore Kanban refers to "signal cards“
- Starbucks is a Kanban-System -> the cup is the Kanban
- The cup-ban acts as an order form that can encode
  most combinations that a barista should expect
- SCRUM-Boards are often traditional Kanban-Boards

So… what is the problem?

Pitfalls

- What are potential problems/pitfalls of agile methods?
- Mostly by the example of SCRUM (as it‘s the most used one)
- BTW: I do not know any game development studio that uses SCRUM 100% by the Book
- Most teams just use parts of it – the once that work for them
- SCRUM Purists call this approach SCRUMBut
- They blame wrong usage when it‘s not working out
- Which I personally think is at least questionable…

The It vs Games Dilemma

- Software Development Processes = traditional IT
- We are not IT Development
- We are not making SAP Databases
- We are making Games
- There are fundamental differences between the two

The IT vs Games Dilemma (cont.)

- The Customer (Publisher) does not make the Design/Specification doc (compared to traditional IT)
- The user (player) does not give feedback until Game is out or focus group tests with Closed Alpha/Beta
- No Design in the World can „describe“ fun
- True: that‘s why an iterative approach for „Making it fun“ is a good concept (Prototypes/Pre-Production)
- But: if you treat game development solely with methods that were meant for creating a database – you easily end up creating just a database, but not a game!

The Deadline Paradoxon

- Working with a Publisher ALWAYS means: There are Deadlines
- Good reasons for that: Planning for Marketing, QA slots, Console/Apple Submission etc.
- Not committing to deadlines does not work with Publishers
- SCRUM Purist argue the product is potentially shippable after *every* Sprint – in theory that would mean:
  - Just stop developing at the deadline and release the game, no matter what
  - In whatever state it is at the end of the last Sprint
  - The goal of game development is not "to finish a game" but "to finish a *good* game"

The Feature Problem

- A feature may take longer than just one sprint
- Many larger features don't work with the concept of being 'potentially shippable' after every Sprint.
- Sometimes there are features you either do entirely in one go or not at all, and that may just take the time it needs and can not be squeezed into a sprint
- Other problem: a large feature may only work together reasonably with another feature linked to it
- Both have to be there to make it meaningful. Half a feature-set might be just a broken game

The Sum Of All Features

- A game is more than the sum of ist features
- SCRUM encourages for divding devlopement into little chunks
- This often ends up in a gathering of working features – without any soul or meaning
- It takes a vision and an overall plan how these features shall work together
- I‘ve seen many SCRUM teams that were just thinking about the current sprint
- It was even kind of „forbidden“ to them to think about the next sprint or upcoming features

The Sum Of All Features (cont.)

- Product backlog: not a good tool to transport an overall vision (nor a good vision for features itself)
- This does not work for Game Development
- It‘s like shooting a movie and on the set you start writing the script, scene by scene while you shoot
- 1st rule of Hollywood Script-Writing: You start developing a script with the end in your mind (= the vision)

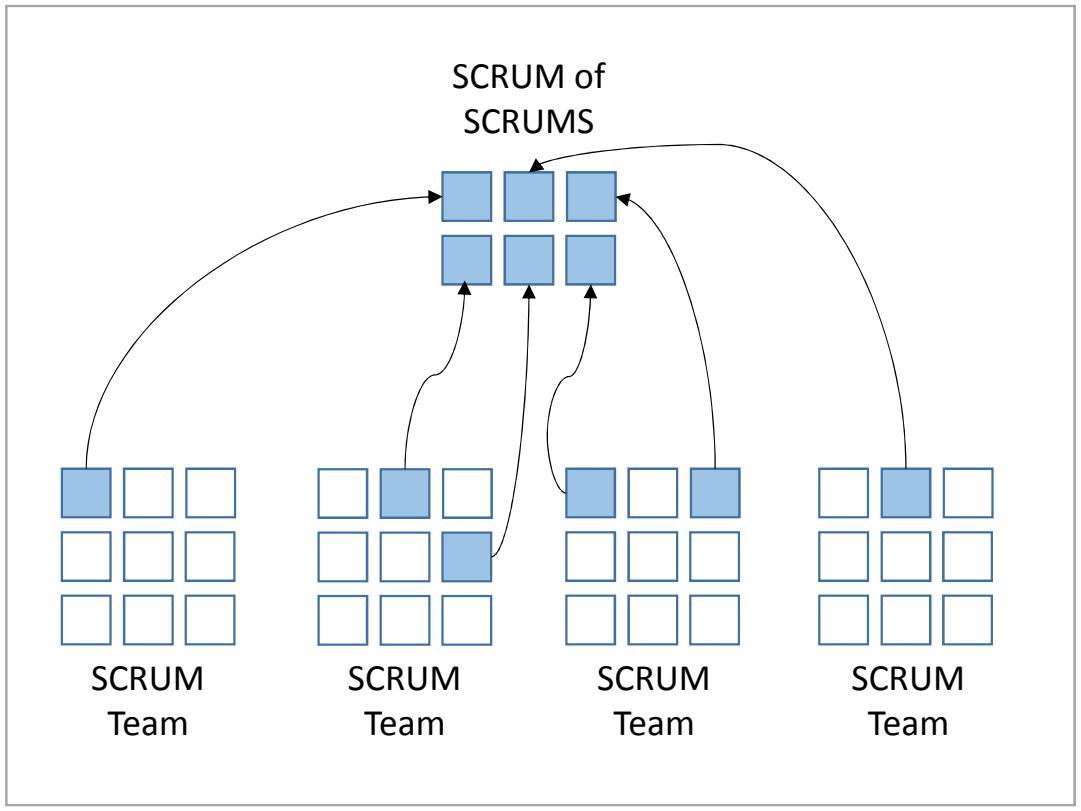
The Daily Stand-Up Charade

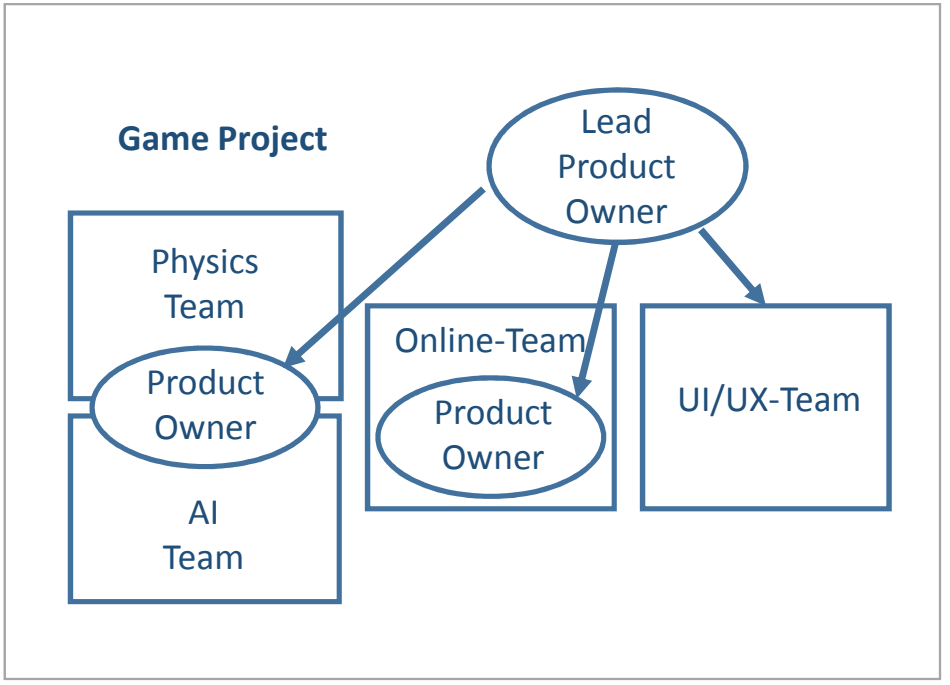
- Generally: Daily Standup is a good thing
- But can easily go wrong and derail
- Lot of people don‘t want to talk there, others talk too much (usually always the same ones)
- SCRUM Stand-Ups can create an interrogation atmosphere
- What if people are not in the same location?
- Question most teams forget in Stand-Up: What‘s the status for the Outsourcing teams/partners?

The Vision-Keeper Complication

- Who owns the Game Vision?
- Problem in many SCRUM teams: missing or unclear Vision Keeper
- Or: Vision Keeper is NOT the Product Owner
- Or: Product Owner is NOT the Game Designer but a „Suit“
- SCRUM Team = all team members have the same saying
- Does hardly work in Game Development (or similar creative processes/media)
- Again: Try to imagine SCRUM on a movie set
- US TV series: Showrunner Princicple in combination with Writer‘s Room

The Scaling Nightmare





The Scaling Nightmare (cont.)

- SCRUM recommends team has size of 7 to 9
- Less you don‘t need this structure
- More you can not let it run „self managed“
- But what‘s with bigger Game Teams/Studios?
- Then suddenly SCRUM gets very hierarchical very fast (see slide before)

The Scaling Nightmare (cont.)

- SCRUM of SCRUMS (somebody in a blog called it „the old idea of communists‘ cells (Russian Bolsheviks organized themselves in “cells” of about 10 to hide from the management of the day)
- Results in LOTS of meetings, makes decision making super-slow
- To quote another blog: “In a nutshell Scrum can add unnecessary overhead & can encourage product management anti-patterns that lead to bad decisions, frustration and a deceiving sense of control.”

The Customer Question

- Most agile methods preach user & customer involvement throughout the whole development cycle
- Who's the user and who's the customer in Game Development?
- Unlike e.g. SAP we can't put UI framework/mask out and ask users (=gamers) to give feedback
- Does hardly work for games without context
- Publisher also can't be involved like traditional Product Owners
- Publisher usually has no time for such an involvement, nor the expertise (the publisher producer might have, but not all other stakeholders)

The Burndown Chart Complication

- Goal of Burn-down Chart: To measure Velocity during Sprint
- But…
  - Premise: always the same people
  - Premise: that always work the same speed
- So Question is: What do you actually measure?
- How good/fast the people work?
- Or how good their estimates were?
- My Experience: Burn-Down Chart one of the first things most teams start to not update anymore once they are running out of time

The „Individuals Over Processes“ Lie

»Individuals and interactions over processes & tools«

1st core from the Manifesto for Agile Software Development (2001)

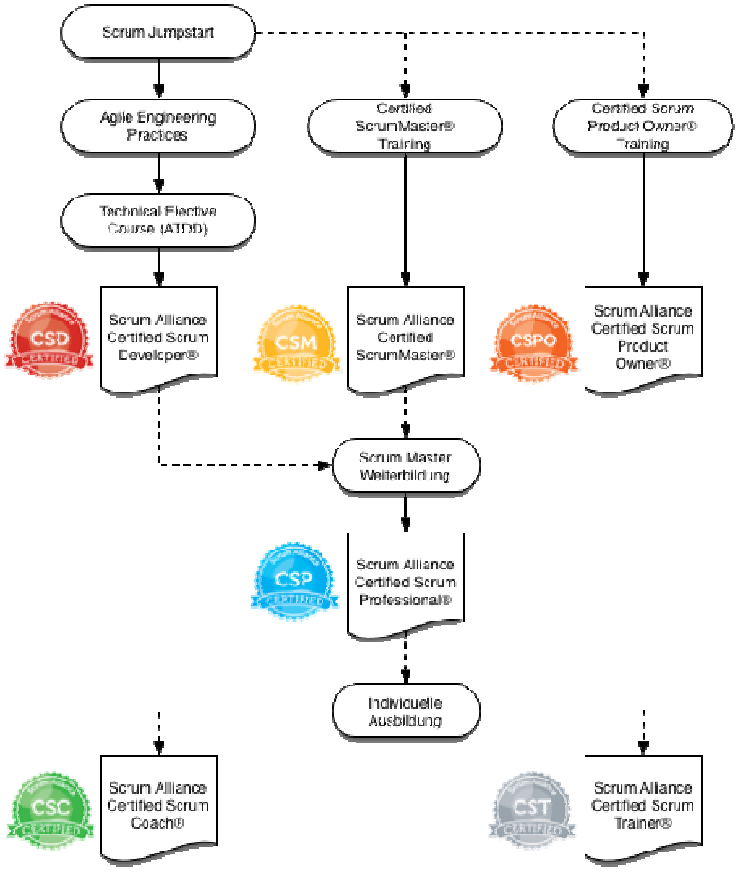
VS

»ScrumButs are reasons why teams can’t take full advantage of Scrum to solve their problems and realize the full benefits of product development using Scrum. Every Scrum role, rule, and timebox is designed to provide the desired benefits and address predictable recurring problems.«

SCRUM.ORG

THE „INDIVIDUALS OVER PROCESSES“ LIE (CONT.)

- Typical SCRUM statement:
- *“You don’t see good results with Scrum because you don’t do enough of it/don‘t use it right!”*
- Such statements encourage the opposite of the (very pragmatic) principles of the original Agile manifesto (see slide before)
- Reason: It‘s all about certificates (=MONEY!)
- Every two years you have to renew your SCRUM Master/Product Owner etc.



Some Best Practices…

Vision Statement

- Vision Statement is fundamental…
- …even when using agile methods
- If you do not know the goal you will not find the way
  - What is it that you want to build?
  - What will be the core gameloop the player will constantly face & play?
  - What are the primary and secondary gameplay elements?
- It‘s hard to „iterate“ towards a vision – how do you know when you‘re there?

The „Hitchcock Approach“

- Always go from the big Picture into smaller pieces (like Hitchcock)
- Don’t start creating a Backlog without clear a Vision Document
- Don’t start creating small feature implementations if you do not know the big picture
- Actually the basic idea of starting with “Epics” and then breaking them down into User Stories is pretty good and exactly the right approach in that context
- Just make sure you have the big Picture first, before you start writing User Stories (potential downside of “Agile methods as they come from IT: you start with implementation of stuff that you think is clearly enough defined already)

Pre-Production

- Never start without a proper Pre-Production
- Topic big enough for another talk in iteself
- Briefly: spend rather too much time in Pre-Prod than too little (upstream vs downstream cost)
- One of the many advantages: Team is smaller
- Pre-Production itself well suited for agile methods as it is per definition about prototyping and iteration

VERTICAL-SLICE

- Final Proof of production quality and fun
- Important when working with a Publisher
- Final alignment on what it is that the Publisher (and ultimately the player) gets in the end in terms of quality level and fun

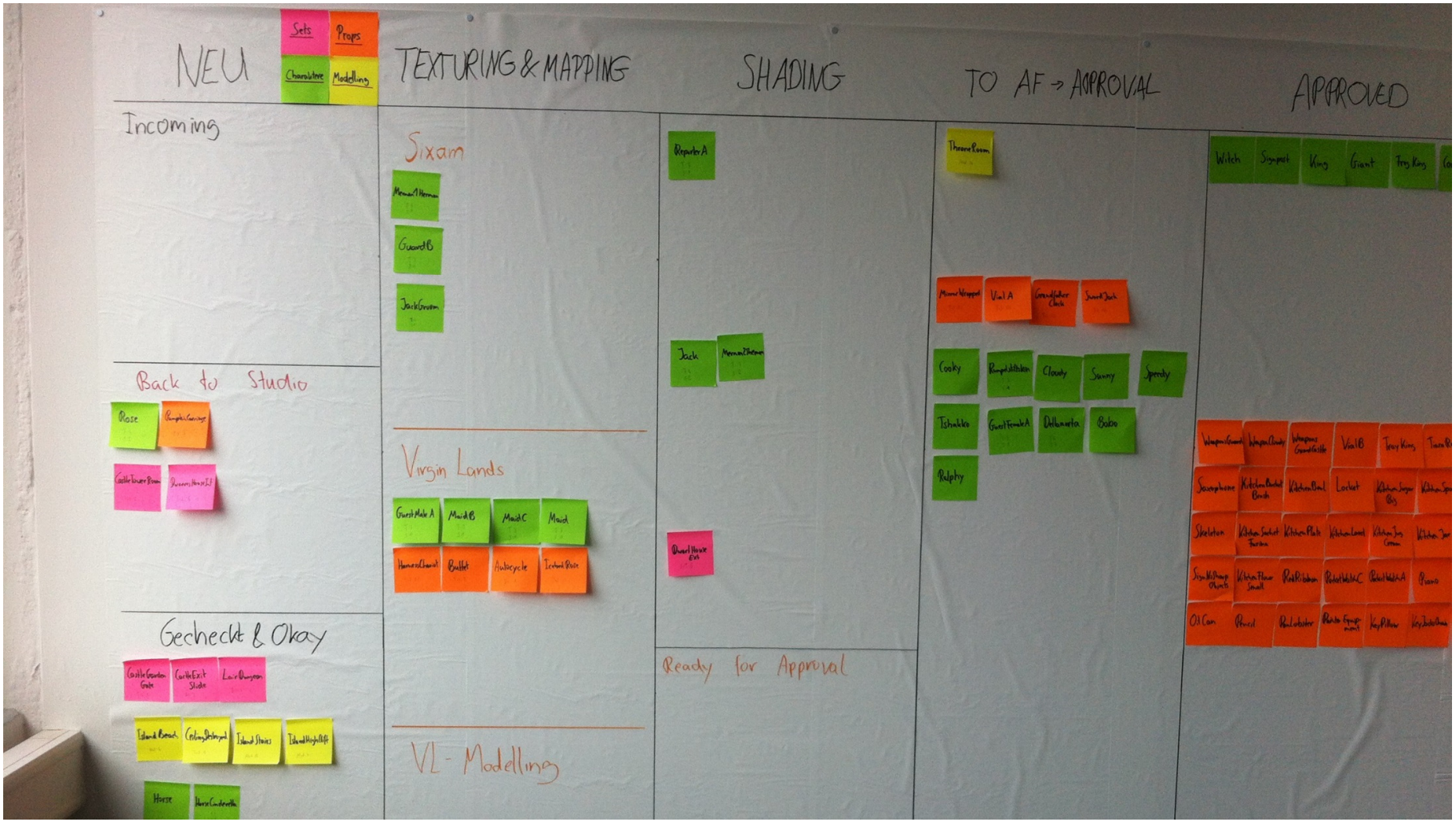
Feature-Creep

- American Game Design vs. German Game Design:
  - US-Design: Feature is not fun? It gets kicked out…
  - German-Design: Feature is not fun? Add two additional features to support the feature that is not workin
- Create a small feature set that is deeply polished – not the other way round
- Once again: potantial trap of Agile-Methods as they preach „Delivery of Features“
- I‘ve seen too many teams (and publishers!) that were only focussed on creating features rather than fun

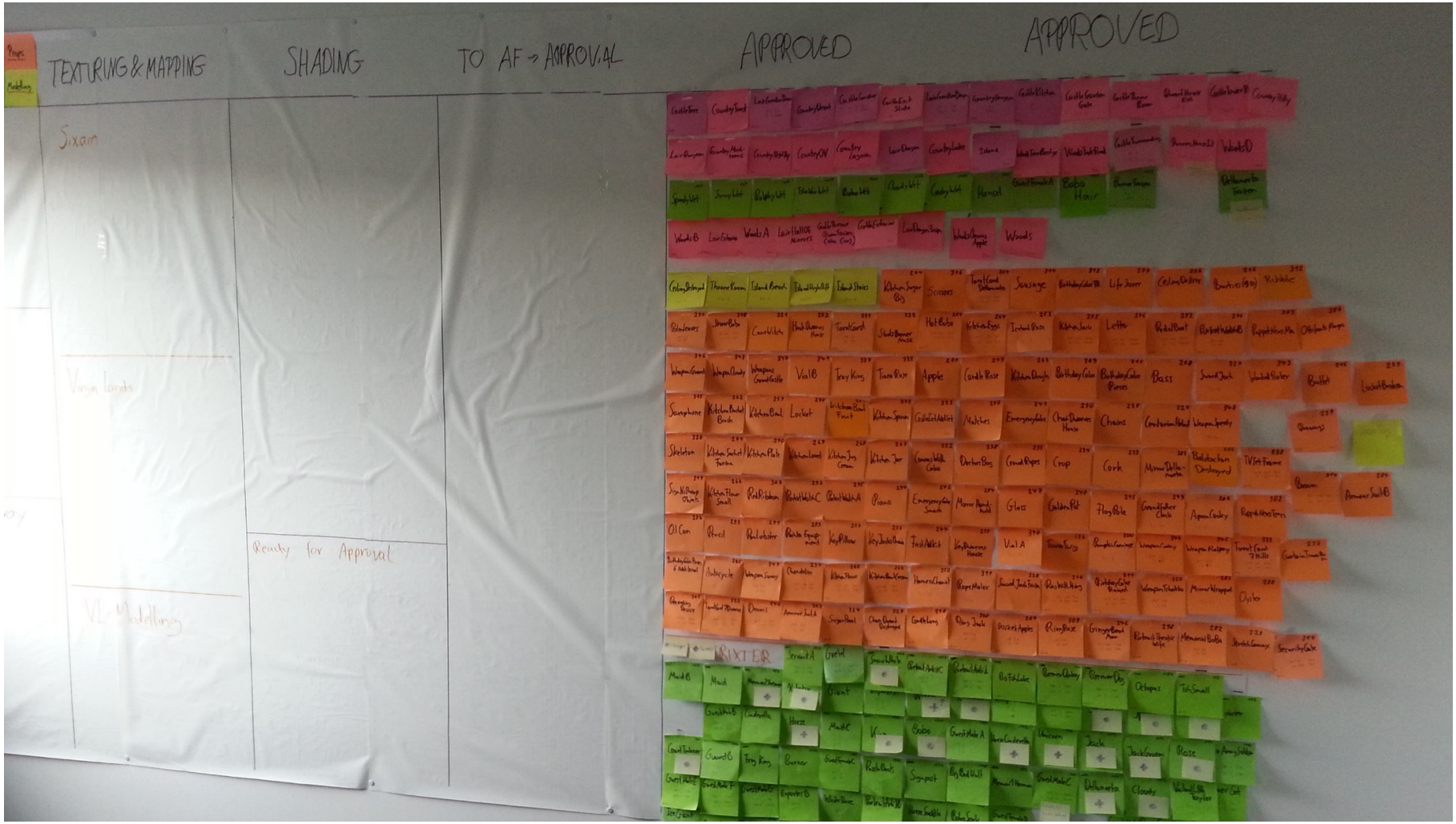
CANBAN-BOARDS

- Kanban Boards (e.g. Trello) are great and can be used and adapted for project management in multiple ways
- Next Slides show how we used it at Virgin Lands for instance to plan an entire Movie Production
- In this example: Texturing Workflow of all Movie Assets
- Kanban generally is very helpful especially for creation and tracking of graphic Assets
- Most Movie Production use a tool called „Shotgun“

CANBAN-BOARDS



CANBAN-BOARDS



There Is No „Too Simple“

- With regards to Project Management Methods…
- …listen to Albert Einstein*: Make it as simple as possible but not simpler!*
- It’s not about creating a Leviathan-Process!
- Trust your instincts – if you think your method sucks, than it most certainly does!

Team Acceptance

- Last but not least: acceptance of the methods used is more important than their potential efficency or any dogmatic framework
- Tailoring your process to your situation has always been the most agile option of all
- The whole team should agree to any process and workflow
- If the team thinks a workflow is stupid – change it!
- They will not follow it otherwise anyway