




An Agile Retrospective

Clinton Keith
CTO, High Moon Studios




WWW.GDCONF.COM



Overview

The goal of this session is to address how to adapt agile practices for game development while not destroying the values and principles behind them.

- Retrospective format
 - What works (clear wins)?
 - What doesn't work so well?
 - What do we need to start doing?
- Info gathered from developers
- A broad range of topics, not in depth
- Talent and leadership still # 1



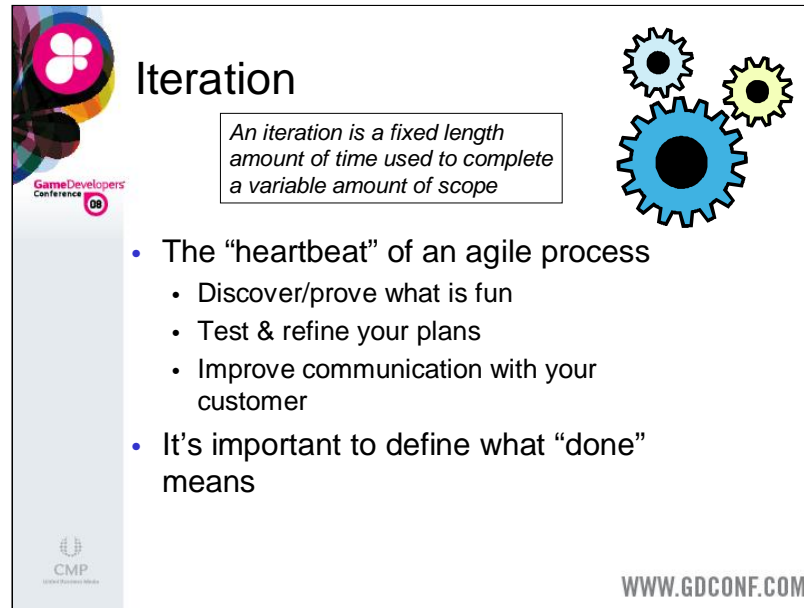
WWW.GDCONF.COM

This presentation follows the team retrospective format. A retrospective is in-progress post-mortem that addresses what works, what doesn't work so well and what things we need to start doing. Although retrospectives are usually an open forum, we can't manage that with a group this size. However much of this information has been collected from not only High Moon teams but many teams in the industry that have been using agile for years.

I also want to reiterate that we're not talking about the "only way to make games". Talent and leadership are the number one factors in determining the quality of a game. Agile can help clear the path a bit.



The first thing we'll address are the things that work for us.



Iteration

An iteration is a fixed length amount of time used to complete a variable amount of scope

- The “heartbeat” of an agile process
 - Discover/prove what is fun
 - Test & refine your plans
 - Improve communication with your customer
- It’s important to define what “done” means


Game Developers Conference 2008

CMP

WWW.GDCONF.COM

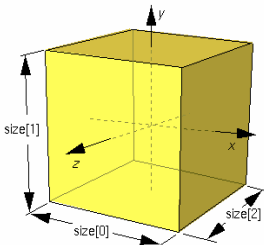
Iterations are the foundation of any agile approach. It’s the heartbeat of the project that tells us what is fun and what isn’t. We use iterations to iterate on the plan and improve our communication of the vision of the project with our customers.

It’s important to have a clear communication with the team on what we mean by “done”. There can be many levels of “done” that are used for the goal of an iteration. You don’t want to short-change yourself by showing 90% of something done. On many problems, there is the “first 90%” and the “second 90%” which reflects how much risk there is in that last few bits that need to make something truly done.



Time-boxing Art

A time-box is a fixed length of time given to produce results. The results are variable.



- "Perfect is the enemy of good enough"
- Quality is a variable that the customer should judge based on cost
- Especially good in production, but need to fine tune timebox for quality & improvements
- Best for "macro level" assets
- Scarcity drives innovation

CMP
Creative Management Practice

WWW.GDCONF.COM

We found that timeboxing some of the artwork is a valuable tool for a team. While it might not be fully agile to tell an artist that they only have fewer hours than they hope for an asset, it has many advantages.

Too many times an artist will produce something of higher quality than the customer needs. For example, they may want to spend a couple of days creating a fire hydrant. While this might produce the best fire hydrant you've seen, it's not worth the cost based on the need of the game. You might pass that fire hydrant at 90 MPH and not care about the extra attention. As a customer you can demand that no more than X hours are spent on the fire hydrant. That timebox may need to be adjusted based on quality. For example if you gave them 30 minutes to do it, you might not like the result.


I don't mean to pick on artists here, but result is that you get innovation from scarcity. If an artist has less than the ideal amount of time they will be forced to drop the quality at first, but also start to find ways to improve the product given the time they have. This doesn't happen with unlimited time.



Agile and Leadership

- Creating ownership
- Unity of vision on large teams
- Leading by doing

The role of leadership in an agile culture shift to a mentoring/support role from a command and control role

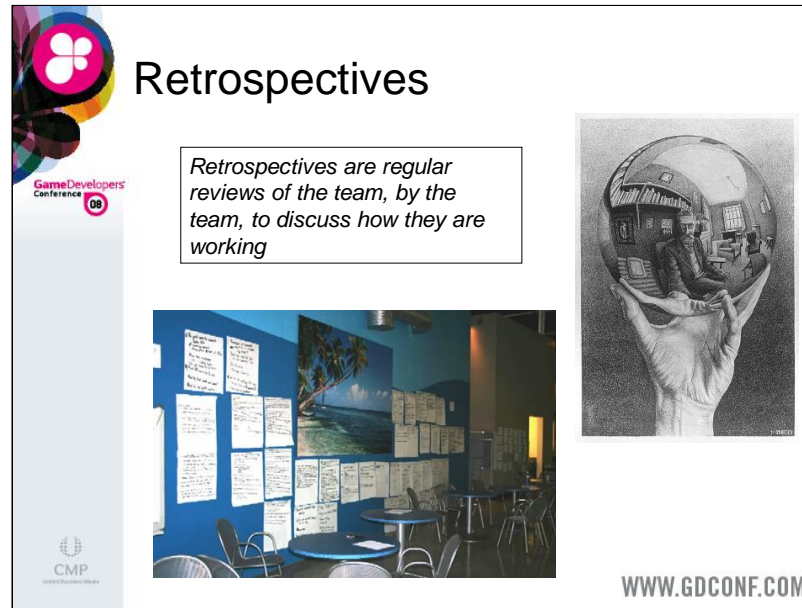


CMP
Creative Media Partners

WWW.GDCONF.COM

When we started using Scrum we thought it would deemphasize the lead role on the teams, but that wasn't true. It shifted the role from a command-and-control approach to a mentoring role. Their role is focused on how to do their work rather than what to do. They also help create a unity of vision throughout the team. For example, if the AI programmers are spread out on different teams, the role of the lead programmer is to insure that they are not repeating effort or working at odds. In fact our lead programmers form programmer scrums that meet once a week to address issues.

This frees up a great deal of the lead's time which would formally have been spent estimating and tracking. The benefit is huge since the lead programmer can lead by doing which exposes them to real issues.



The slide features a title 'Retrospectives' in a large, bold font. To the left is a logo for the Game Developers Conference (GDC) with the text 'Game Developers Conference' and 'GDC'. Below the logo is the CMP logo. A central text box contains the definition: 'Retrospectives are regular reviews of the team, by the team, to discuss how they are working'. To the right is a black and white illustration of a hand holding a globe that reflects a room with people. At the bottom right is the website 'WWW.GDCONF.COM'. The bottom left corner has the CMP logo.

Retrospectives

Retrospectives are regular reviews of the team, by the team, to discuss how they are working

WWW.GDCONF.COM

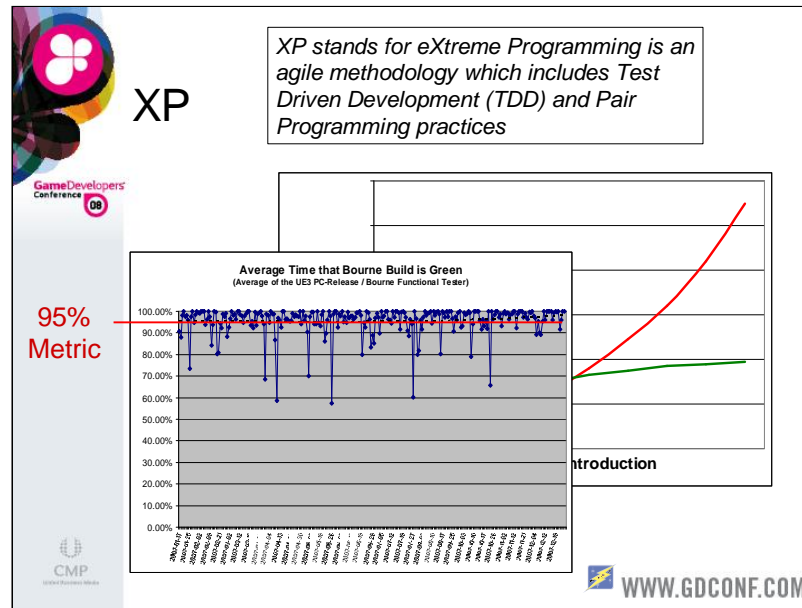
This creates the “inspect and adapt” cycle for how a team works together and alters their practices to improve how they work. Teams get together to answer three questions:

- What do we need to continue doing.
- What didn't work.
- What do we need to start doing.

Valuable at every level

- Team
- Project
- Company

In fact we will have full company retrospectives every six months. It's less formal and we address everything about how we make games and organize the company. Groups self organize to discuss a topic for 45 minutes and come back and present their results to the company. The notes from the meeting cover a 50 foot wall in the kitchen by the end of the day and result in hundreds of action items that improve the company.

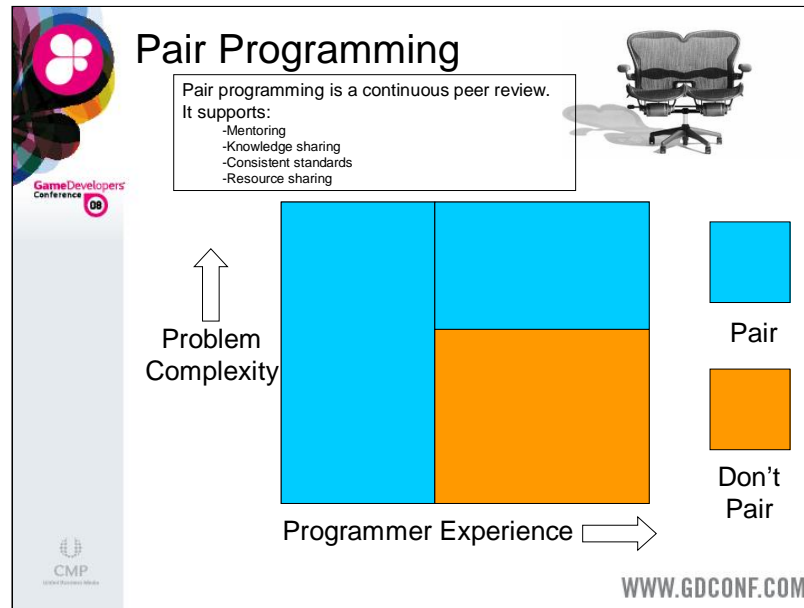


One of the methodologies that has worked for us is XP, or extreme programming. Includes TDD writing tests along with the code and pair programming which I'll talk about in a few slides. The goal of XP is to keep the cost of change down

Waterfall cost of change is higher because proof of the architecture plan doesn't happen until the end which can require major changes like changing a car engine after you have finished building the entire car. The problem with Scrum was that the constant change of requirements created problems with the technology keeping up and staying stable. It's a lot easier to keep a bad design stable than to change the design the architecture was based upon. XP was developed to address the changing needs of any design.

XP is architect-as-you-go: I believe in a mix of some architecture and some refactoring. The key is to architect based on what you know.

We have over 14,000 tests in our code. This is a graph of the stability of the game over 2007 based on tests run on the PC release version. This shows that for an average of 95% of the time, a person working on the game can get the latest build and it should be fully stable. The benefit of this is huge. It allows progress to be made on a steady basis rather than waiting days to see changes propagate through a system or have to wait longer if the build is broken. The tests catch over 90% of the bugs that are introduced.



Pair programming is a continuous peer review. One doesn't sit there while the other types. There are numerous benefits:

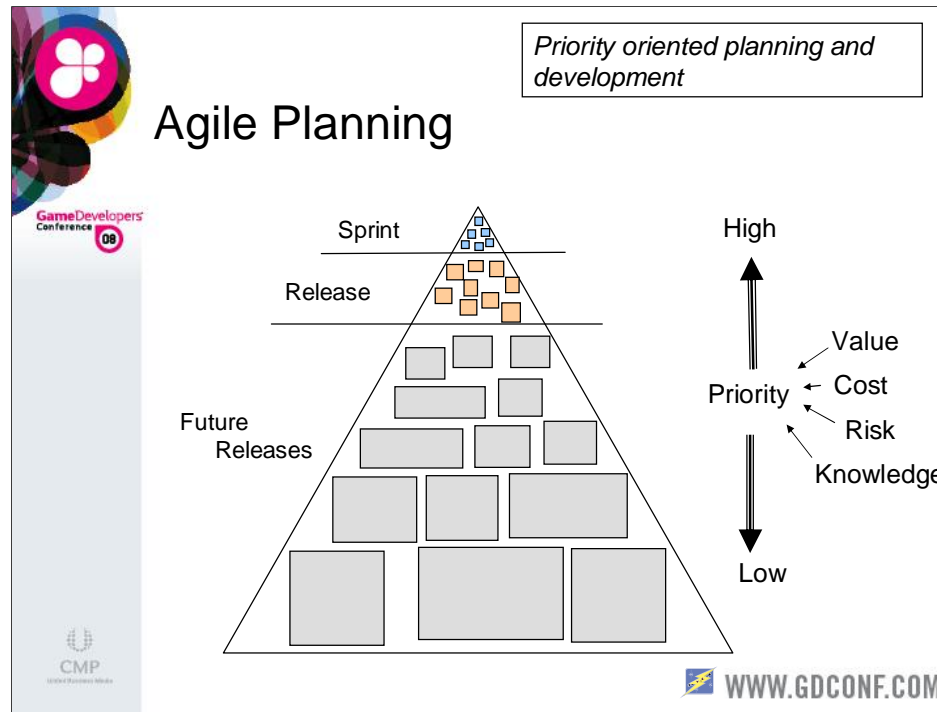
- Mentoring
- Education
- Consistent code base
- More than 1 person knows the code

We don't pair all the time.

- When a new programmer is entering the picture, you pair up. So for example, if a programmer comes over from another team to the driving team, but doesn't know about the driving tech, they would pair up to work on any driving tech for awhile.
- When a problem is complex enough, then pairing up to solve it makes sense. This gets you past the "sticking points" on problems by handing off the keyboard or having frequent discussions about the solution. This also creates a better quality solution/

Parallel programming

- Decide to pair up or work in parallel on similar tasks and peer review consistently



Agile Planning

- Planning = Design (how) + Execution (what)
- You can't plan away uncertainty
 - You have to execute to reduce uncertainty
- Planning is spread out over the entire project
- Shifts the emphasis from “the plan” to planning

- Shifts from “completion of activities” to “delivery of features”
- Creates plans that are easily changed & encourage change
- Plans are focused on *releases* of the game. Releases are “potentially shippable” versions of the game.

A good analogy for Agile planning is to think of your product backlog as an iceberg.

Disaggregation occurs every sprint. The larger, lower priority chunks on the bottom become higher priority and get broken down.



Game Developers
Conference

Coaching & Classes



- Onsite Coaching
- Scrum Master Classes
 - Focused on Game Development
- Product Owner Classes
- Estimating and Planning Classes



WWW.GDCONF.COM



Game Developers
Conference 08

What Hasn't Worked So Well



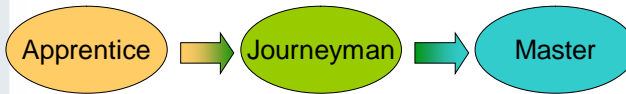
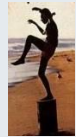
CMP
Creative Media Partners

WWW.GDCONF.COM



Adoption issues

- Scrum is hard
 - Changing practices from the start can backfire
 - ...but change is necessary



- Silver bullet mentality
- XP is controversial
- Leadership support & sabotage



WWW.GDCONF.COM



Game Developers
Conference 2018

Scrum for Artists and Designers

- The shortcomings of Scrum
 - Real flow is more complex
 - Specialists vs. generalists
- There are no XP-like practices for artists and designers
- Lean and Kanban may provide some answers



WWW.GDCONF.COM



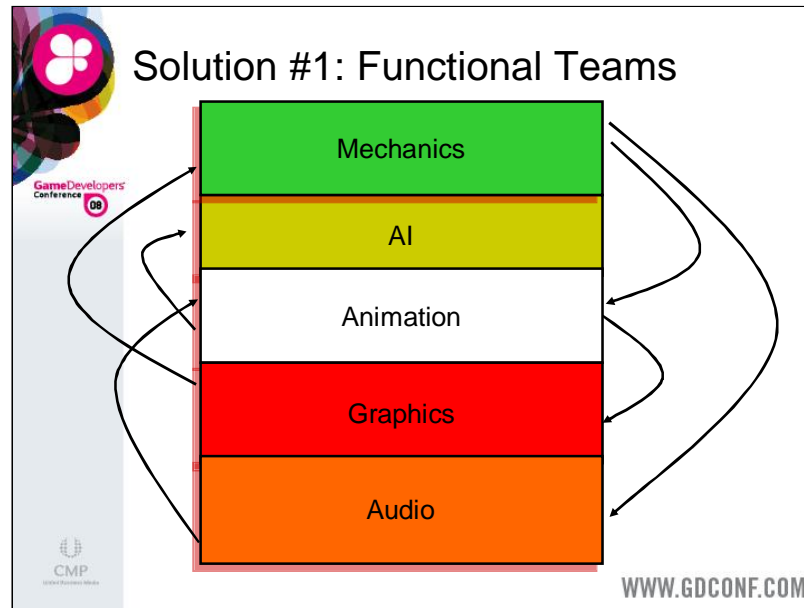
Game Developers
Conference

Large teams

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



WWW.GDCONF.COM



Solution #1 : Functional Teams

Example: Artificial Intelligence

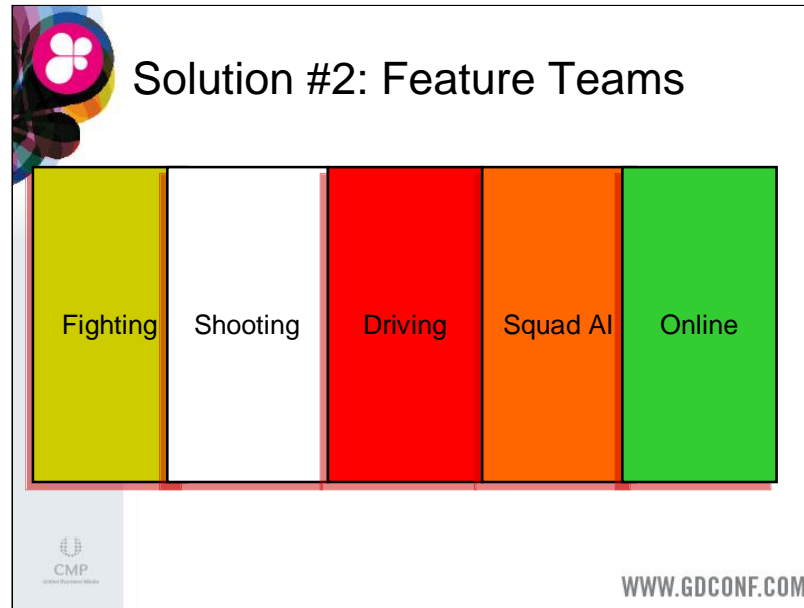
Every other team needs it

Concentrate the AI experts on the team

Functional teams didn't work

No real “product focus” resulted in making a really great architecture that impressed other AI programmers.

Interdependencies create delays of at least a sprint in seeing real value



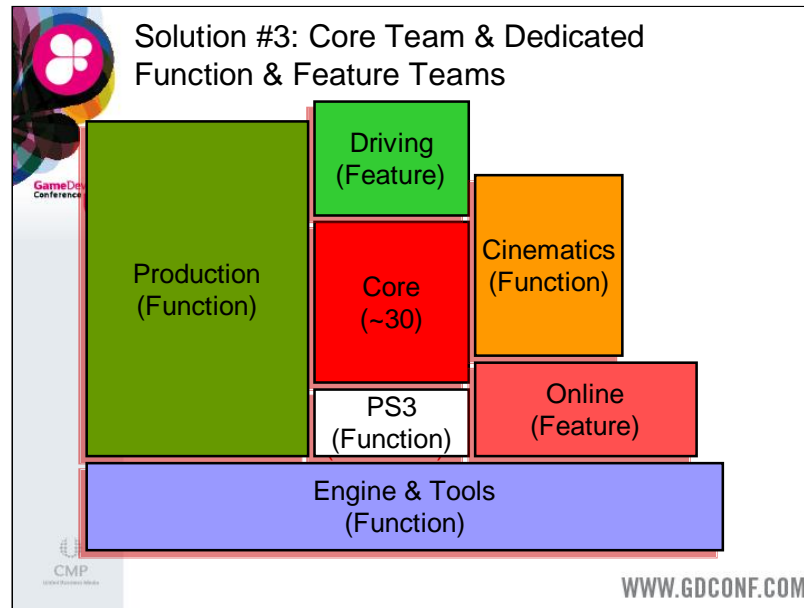
Solution #2 : Feature teams

Ever release the project team would self organize into teams that would focus on one feature per Sprint

Problems :

Teams change so frequently that ownership is a problem

Ownership across features are equal and competitive. This harms a “one product vision”




Solution #3: A mixed approach

- Some teams are functional
- Some teams are feature
- Core team owns the product vision
- Other teams are dedicated to features across products




Benefits:

- Creates ownership locally
- Minimizes frequent reallocation across different teams
- Promotes building great teams with pride and experience
- Plugs in outsourcing and lean production into a agile driven product



Fears of Agile Planning

- “Agile planning means no planning”
- “Creates endless iteration”
- “Can’t be used for production”
- Agile vs. Waterfall, a polarization of views



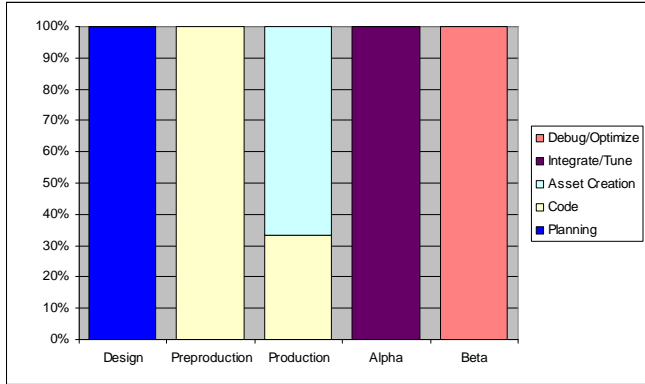
WWW.GDCONF.COM

Agile planning is more planning intensive because we don't just “get it out of the way”



Ideal Waterfall

Game Developers Conference 08



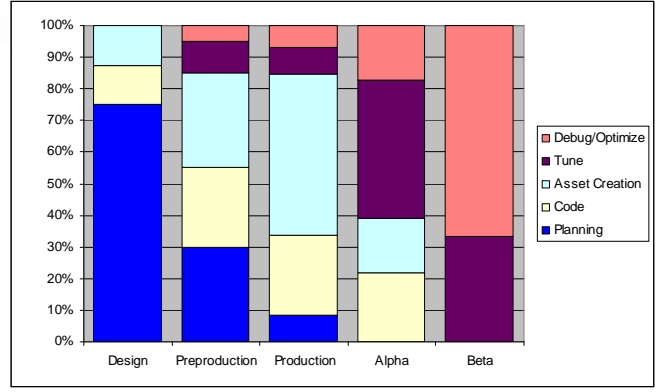
CMP
Creative Media Partners

WWW.GDCONF.COM



Game Developers Conference 08

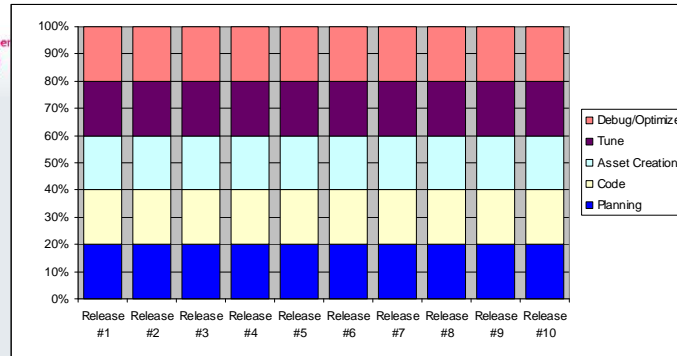
Actual Waterfall



WWW.GDCONF.COM



Ideal Agile

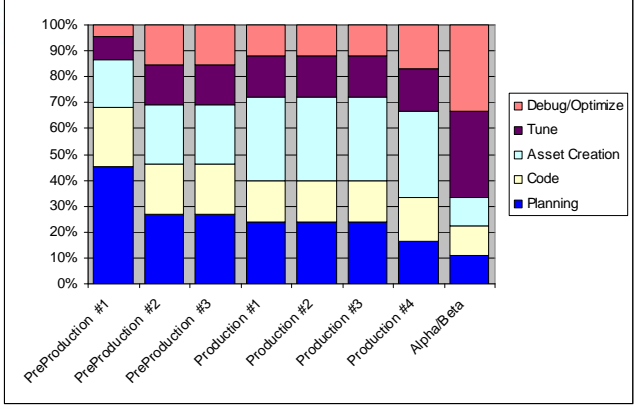


WWW.GDCONF.COM



Agile Game Development

Game Developers Conference 08



WWW.GDCONF.COM



Game Developers
Conference 08

What we need to start doing (new/more/better)



WWW.GDCONF.COM



Agile Transition Strategies

Bottom Up or Top Down?



- **Beachhead team**
 - Low cost & risk
 - Takes more time
 - How to spread?
 - Creates influence
 - Easier to adopt and try all practices
 - Can be a stealth skunk works project
- **Entire company**
 - Requires more coaching
 - Takes less time
 - More cost & risk
 - Usually requires command and control



WWW.GDCONF.COM





Game Developers
Conference 08

Art Production & Agility

- Production and content creation aren't a perfect fit with Scrum
 - Complex flow of work
 - Cross-discipline teams
 - Better understanding of requirements

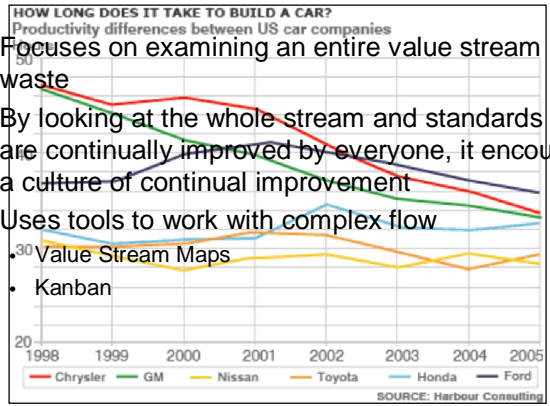


WWW.GDCONF.COM

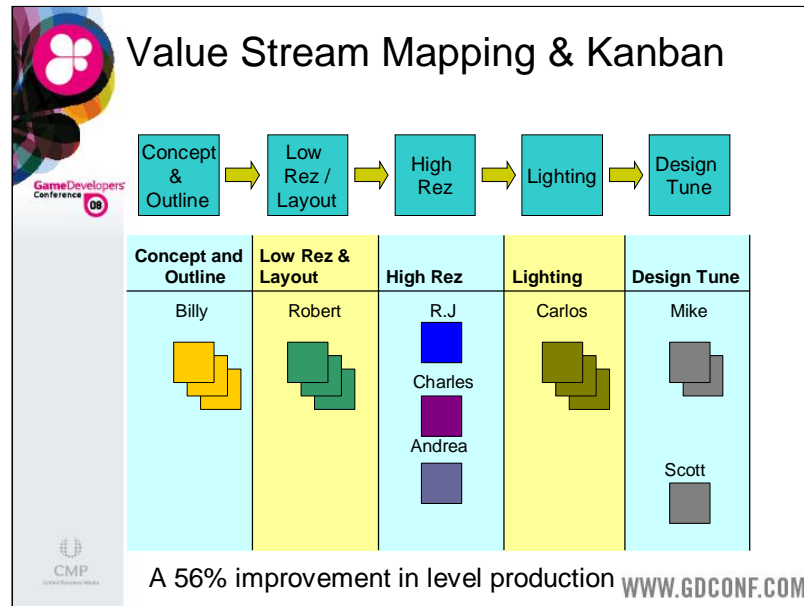


Lean Production

- Focuses on examining an entire value stream and waste
- By looking at the whole stream and standards that are continually improved by everyone, it encourages a culture of continual improvement
- Uses tools to work with complex flow
 - Value Stream Maps
 - Kanban



WWW.GDCONF.COM



Focus on reducing waste

- Handoffs
- Waiting
- Reducing cycle time
- Task Switching



Game Developers
Conference

Conclusion

- Principles over practices
- Metrics are key
- Talent & Leadership are #1
 - (People over process)
- More info
 - www.AgileGameDevelopment.com
 - www.MountainGoatSoftware.com
 - CSM-VGD in May

Questions?



WWW.GDCONF.COM