Agile Game Development
Dealing with Chaos in the Real World

Noel Llopis
Lead Technical Architect
Sammy Studios

nllopis@sammystudios.com
http://www.gamesfromwithin.com

3 November 2004

What is this talk about?

- Agile development applied to game development.
- Why? Because agile development is a good alternative to most development methodologies used in the games industry.
- We'll talk about:
  - Part I: Risk and methodologies
  - Part II: Agile development
  - Part III: Scrum
  - Part IV: Extreme programming

PART I

Risk and Methodologies

What's a Methodology?

A development methodology is how a company chooses to organize people and resources to develop its projects.

Risk and Methodologies

- Methodologies aim to reduce the amount of risk in a project.
- There isn't a single best one, it depends on the project and the team.

What are we afraid of in the games industry?

What are we afraid of in the games industry?

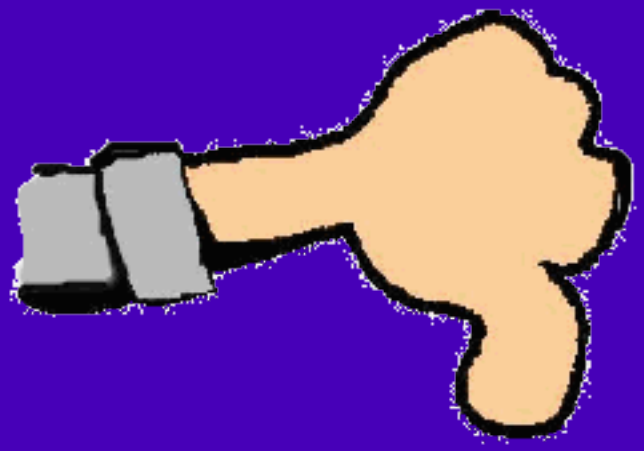
- Schedule slips (oops, we missed Christmas)

What are we afraid of in the games industry?

- Schedule slips (oops, we missed Christmas)
- Project is cancelled (company goes under?)

What are we afraid of in the games industry?

- Schedule slips (oops, we missed Christmas)
- Project is cancelled (company goes under?)
- Game is no fun (bad sales and reviews)



What are we afraid of in the games industry?

- Schedule slips (oops, we missed Christmas)
- Project is cancelled (company goes under?)
- Game is no fun (bad sales and reviews)
- Game has technical problems or is obsolete technically (bad sales and reviews)

What are we afraid of in the games industry?

- Schedule slips (oops, we missed Ch
- Project is cancelled (company go
- Game is no fun (bad sales and revi
- Game has technical problems technically (bad sales and reviews)
- Market changes (our game becomes obsolete)

What are we afraid of in the games industry?

- Schedule slips (oops, we missed Christmas)
- Project is cancelled (company
- Game is no fun (bad sales and
- Game has technical proble
  technically (bad sales and reviev
- Market changes (our game bec
- Staff turnover (delays game even further)

A familiar tale?

- Game design is completed
- Contract with publisher is written in stone
- Detailed schedule is created with dependencies
- Detailed technical design is created
- First few milestones go according to plan
- And then...

A familiar tale?

- You hit technical difficulties that move back the whole schedule...
- or key people in the team leave...
- or a game that redefines your genre comes out (GTA3)...
- or your publisher decides it wants something different...
- or <fill in your last situation here>.

A familiar tale?

What happens next?

- Schedule trashing
- Crunch time. Pressure.
- Bugs introduced left and right. Quality decreases.
- People are burned out and leave.
- ....

It's a vicious circle.

Methodologies: Ad-hoc

- Does it really address anything?
- Low process overhead.
- Low cost.
- Get something working right away.
- Allows very independent/different personalities to "work" however they want.

Methodologies: Waterfall

- Minimizes changes later in the project.
- Tries to deliver exactly what the customer originally asked for.
- Identifies critical paths and attempts to predict delivery date.
- Maximizes visibility and tracking.
- Comprehensive documentation and paper trail.
- Maximizes "efficiency".

Methodologies: Iterative

- RUP, Evo
- Minimizes risk of not being able to deliver a product by having a full deliverable per iteration.
- Well defined architecture.
- Sounds like a lot of game development, but is it really?

PART II

Agile Development

Agile methodologies

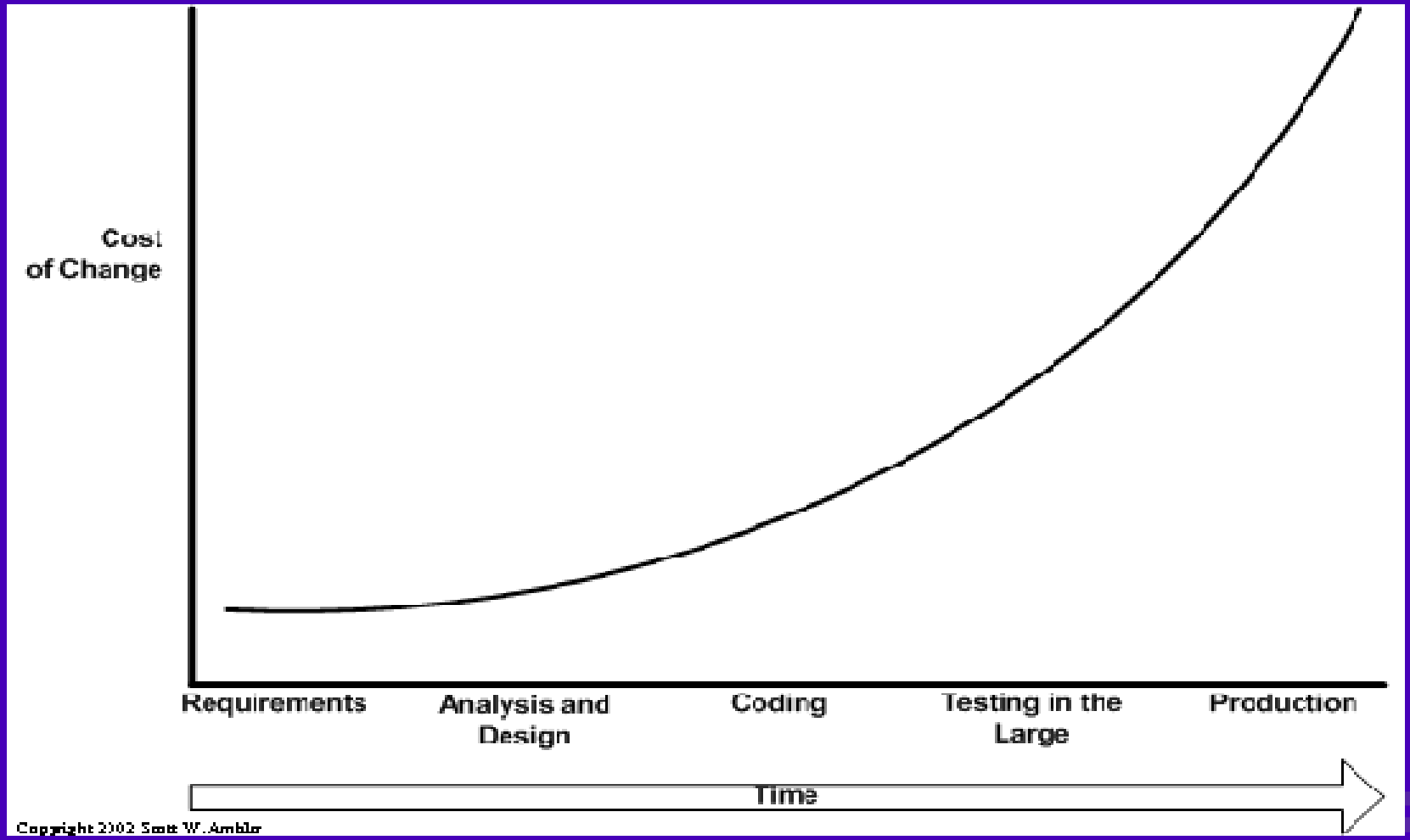
There are many out there:

- Crystal
- Adaptive software development
- Dynamic solution delivery model
- Feature-driven development

and of course, extreme programming and scrum.

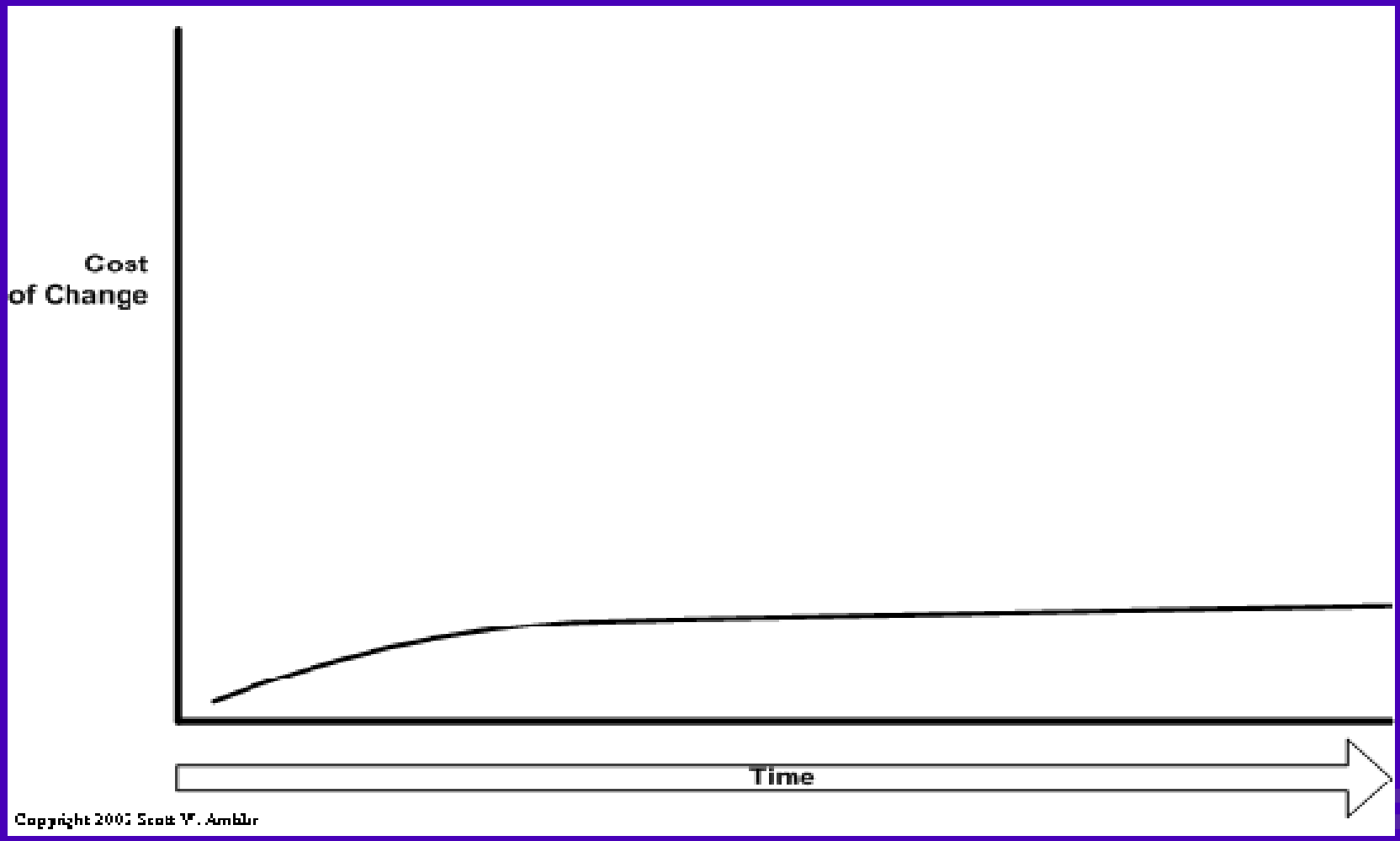
Fundamental idea of agile development

You're probably familiar with this diagram:



Fundamental idea of agile development

Agile development proposes this instead:



What are agile methods based on?

- Individuals and interactions (over processes and tools).
- Working software (over comprehensive documentation).
- Customer collaboration (over contract negotiation).
- Responding to change (over following a plan).

Agile Development

Features

- Individuals and interactions
- Working software
- Customer collaboration
- Responding to change

Game Dev Fears

- Schedule slips
- Project is cancelled
- Game is no fun
- Game has technical problems
- Market changes
- Staff turnover

How is it different?

Traditional: We don't know what's out there, so we should research it as much as possible and plan for all possibilities.

Agile: We'll concentrate on what we have in front of us, and make decisions as we go based on what we learned so far.

How is it possible?

Rapid feedback.

Don't wait for a postmortem. Get feedback right away at different levels.

- Unit tests (few times a minute)
- Builds and functional tests (few times an hour)
- Task tracking/daily meeting (once a day)
- Timeboxed iteration (once every few weeks)

It's like planning a roadtrip.

PART III

Scrum

What is scrum?

Not this!

What is scrum?

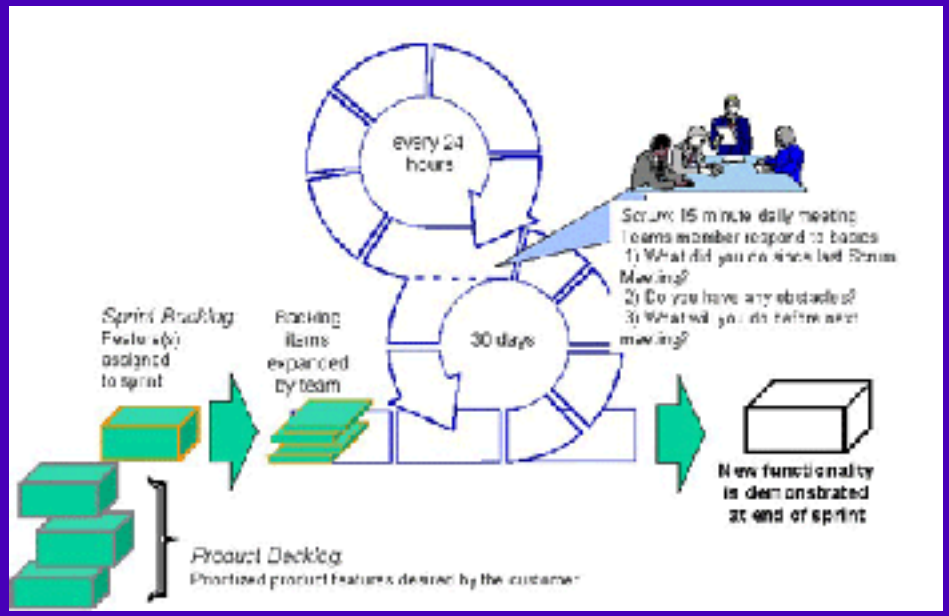
Scrum is an agile, lightweight process that can be used to manage and control software and product development using iterative, incremental practices.

Important: It's a **management** approach.

It can be combined with any iterative development approach.

A scrum iteration

- Product backlog
- 30-day sprint
- Prioritization of goals
- Estimation of tasks
- Self-organizing teams
- Daily team measurement
- End of sprint review



Who's using scrum for games?

Sammy Studios

We have been using Scrum at Sammy Studios (http://sammystudios.com/) for about six months in the development of Darkwatch (PS2/Xbox action title).

Who's using scrum for games?

Also used in Awesome Studios (http://www.awesome.uk.com/) in the UK.

Any others? Email me to let me know.

Scrum and game development

Teams are too large.

Scrum works best for teams of 5-8 people.

- Split up the team into smaller subteams.
- Hold a “scrum of scrums”.
- At Sammy Studios we divided the teams into “functional teams”: characters, vehicles, multiplayer, level production, tools, etc.
- Started with only programmers. Now artists and designers involved as well.

Scrum and game development

Who is the customer?

- Lead designer or creative director for the overall project.
- Identify a customer for each subteam.
- We even have “iteration” teams, whose customer are all the other teams.

Scrum and game development

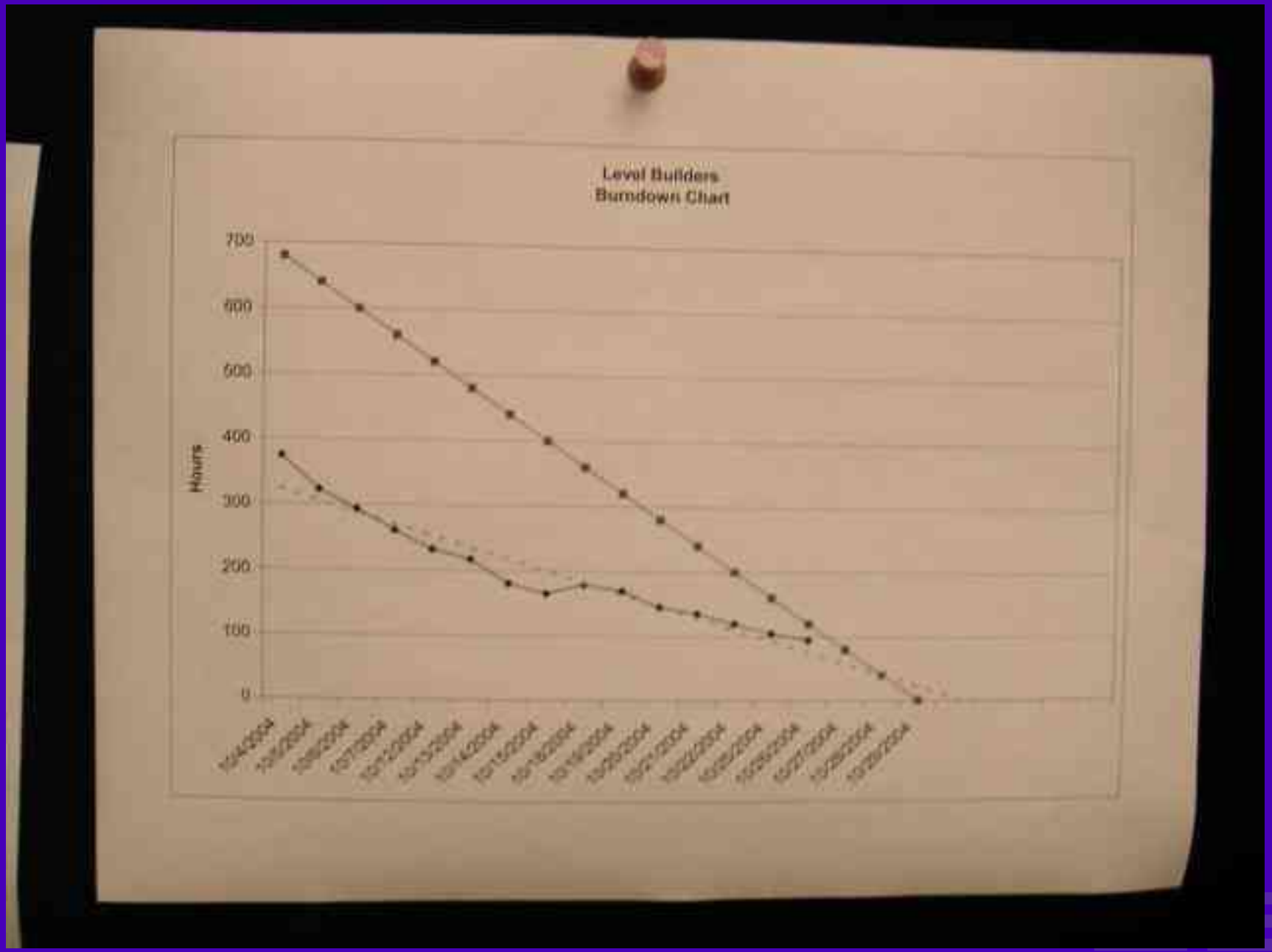
How are bugs dealt with?

- Scrum doesn't say anything about that.
- You can either enter them as backlog tasks.
- Or bring them up as impediments and get them done right away.

The "war room"


LP/MECH/VFX
Sean
Stephane
Joel
Mike

A burn-down chart (daily measurement)



Lessons learned

Scrum will bring up any flaws in your process right away.

- You're iterating very quickly.
- Very good at tracking progress.
- Can measure team “speed”.
- Burndown charts invaluable in seeing progress.
- Estimate _remaining_ hours left in tasks.

Lessons learned

Reporting impediments is crucial, but very difficult.

- Beware if there are no impediments being reported.
- Encourage (or force) everybody to bring them up.
- Lack of impediments probably means lack of buy-in in the part of the teams.
- Typical impediments we hear: nightly build was screwed up again, my machine keeps crashing, some levels are too slow to do any work in them, etc.

Lessons learned

Avoid inter-team dependencies.

- It's sometimes difficult, but avoid inter-team dependencies within one sprint.
- Otherwise it makes it very difficult for a team to re-prioritize tasks or decide how they're going to do them.
- Example: multiplayer team depending on front-end team to add Xbox Live menus.

PART IV

Extreme Programming

+

!= XP

What is extreme programming? (XP)

Extreme Programming is a **discipline** of software development.

It is based on four main values:

- Simplicity
- Communication
- Feedback
- Courage

Core practices

Whole team

Planning game

Small releases

Customer tests

Simple design

Pair programming

Test-driven development

Design improvement

Continuous integration

Collective code ownership

Coding standard

Metaphor

Sustainable pace

What's so eXtreme about it?

If some things are good, then do more of them.

- Code reviews are good? Pair programming all the time.
- Frequent integration? Let's do continuous integration.
- Testing? Let's write unit tests for everything and run them every time we build.

XP practices

The different practices support each other.

- Refactoring needs unit tests.
- Pair programming needs a coding standard.
- Simple design needs refactoring.
- etc...

Beware of adopting only some practices and not others. Make sure they balance each other.

XP and game development

Is it possible? Absolutely!

Game development isn't _that_ different from other types of development.

Amount of binary assets is a big difference.

Code/data interaction can be problematic for rapid functional testing and continuous integration.

XP and game development

Pair programming always controversial.

- We have a lot of specialists in the industry.
- "Prima donnas" don't like pair programming.
- Is the work done pair programming twice the work done by two programmers separately? Not quite, but it's of higher quality.

How does TDD work?

You need to implement a small task that is part of a larger feature.

First you write the simplest test that uses that feature.

The test shouldn't pass or even compile.

Implement the simplest possible code that will make the test pass.

Refactor tests.

Rinse and repeat.

Benefits of TDD

Four huge benefits:

- Confidence to do any refactorings necessary.
- Unit tests will catch any problems right away.
- Tests will serve as a documentation that is never out of date.
- The resulting design will be different (and better!) than what it would have been otherwise.

Benefits of TDD

The quality of a codebase is directly related to how easy it is to refactor it.

The day you don't dare make a change because of what it might cause, the codebase is doomed.

Extensive unit tests prevent that from ever happening.

TDD for games

How can we do test-driven development in game development?

One step at a time. Remember, they are unit tests.

Test most things without involving graphics at all.

Helps to have highly modular libraries so you only pull in what you need.

Doing TDD

Use a unit-testing framework.

- It should be really easy to add new tests.
- Tests should be built and executed as soon as any code is compiled.
- Tests should run very quickly.
- For C++, check out CppUnitLite (also have a look at CppUnit and Boost Test Framework).

Let's do TDD in a game

Task: Walking over a health powerup.

```
TEST (HealthPowerup, AddsHealth) {
    Player player; // Created at the origin
    HealthPowerup powerup; // same here
    int beforeHealth = player.GetHealth();
    player.Update();
    CHECK (player.GetHealth() > beforeHealth);
}
```

TDD in a game

```
TEST (HealthPowerup, AddsCorrectAmountOfHealth) {
    Player player; // Created at the origin
    HealthPowerup powerup; // same here
    int beforeHealth = player.GetHealth();
    player.Update();
    CHECK_INT_EQUALS (player.GetHealth() ==
     beforeHealth + powerup.GetHealthAmount());
}
```

Too much common code: Refactor!

TDD in a game

```
SETUP (HealthPowerup) {
    Player player; // Created at the origin
    HealthPowerup powerup; // same here
    int beforeHealth = player.GetHealth();
    player.Update();
}

TEST (HealthPowerup, AddsHealth) {
    CHECK (player.GetHealth() > beforeHealth);
}
TEST (HealthPowerup, AddsCorrectAmountOfHealth) {
    CHECK_INT_EQUALS (player.GetHealth() ==
     beforeHealth + powerup.GetHealthAmount());
}

TEARDOWN (HealthPowerup) {
}
```

TDD in a game

Check that player doesn't receive more than maximum health.

Check that powerup is removed from the world.

Check that powerup is not applied unless player is nearby.

Etc...

TDD in a game

But what about graphics?

- No need to test at the pixel level for most developers.
- Test the communication between the renderer layer and the hardware.

TDD in a game

What about AI?

These are unit tests, not functional tests.

How do we test that when a character is shot and it's low in health, it runs for cover.

Many different tasks involved. Test them separately.

- When shot, health should go down.
- When health below certain level, flags are turned on.
- When low in health and shot, run for cover mode on.
- When in cover mode, it looks at the right path nodes.

TDD tips

Make sure the tests are built and ran every time you build your code.

Whenever a bug shows up, first write a test that fails because of the bug, then fix the bug.

Don't be surprised if anywhere between 50% and 75% of your code is taken up by unit tests. It's not wasted time or code! But it's important that the unit tests be well written and are easy to refactor.

The more code has unit tests, the easier it gets.

Continuous integration

Integrate and build the system many times a day, every time a task is completed.

- This can be trickier in games that are data-driven because asset builds can take a long time.
- You want to verify that your changes work before you commit them. Unit tests are a start, but probably need some simple functional tests too.
- This is a great goal to aim for. Development in multiple parallel source control branches is painful.

Wrap Up

Starting with agile development

Resist the temptation to create a custom method from the beginning.

Start with scrum, XP, or whatever you want, and follow it.

Customize it later as you learn more about what works and what doesn't.

Starting with agile development

Getting the publisher on board can be problematic.

- Educate them.
- Involve them as the customer.
- They'll have more say in the project that way, and it'll be better for your project anyway.
- If all fails, look for a publisher that understands agile development.

Further reading

Essential books:

Agile and Iterative Development, Craig Larman

Agile Software Development with Scrum, Ken Schwaber

Extreme Programming Explained, Kent Beck

Other highly recommended books:

Questioning Extreme Programming, Pete McBreen

Refactoring, Martin Fowler

Test Driven Development, Kent Beck

Slack and *Waltzing with Bears*, Tom DeMarco

Further reading

Web sites:

- Control Chaos (http://www.controlchaos.com/). The scrum web site.
- Mountain Goat Software (http://www.mountaingoatsoftware.com/scrum/index.php). More scrum.
- Extreme Programming: A Gentle Introduction (http://www.extremeprogramming.org/)
- Extreme Programming Resources (http://www.xprogramming.com/)

Agile development and games:

- Agile Methodology and Scrum in Game Development. Upcoming GDC 2005 talk by Clinton Keith (Director of Technology at Sammy Studios).

Questions?

You can find these slides at

http://www.gamesfromwithin.com