INFT3960 – Game Production

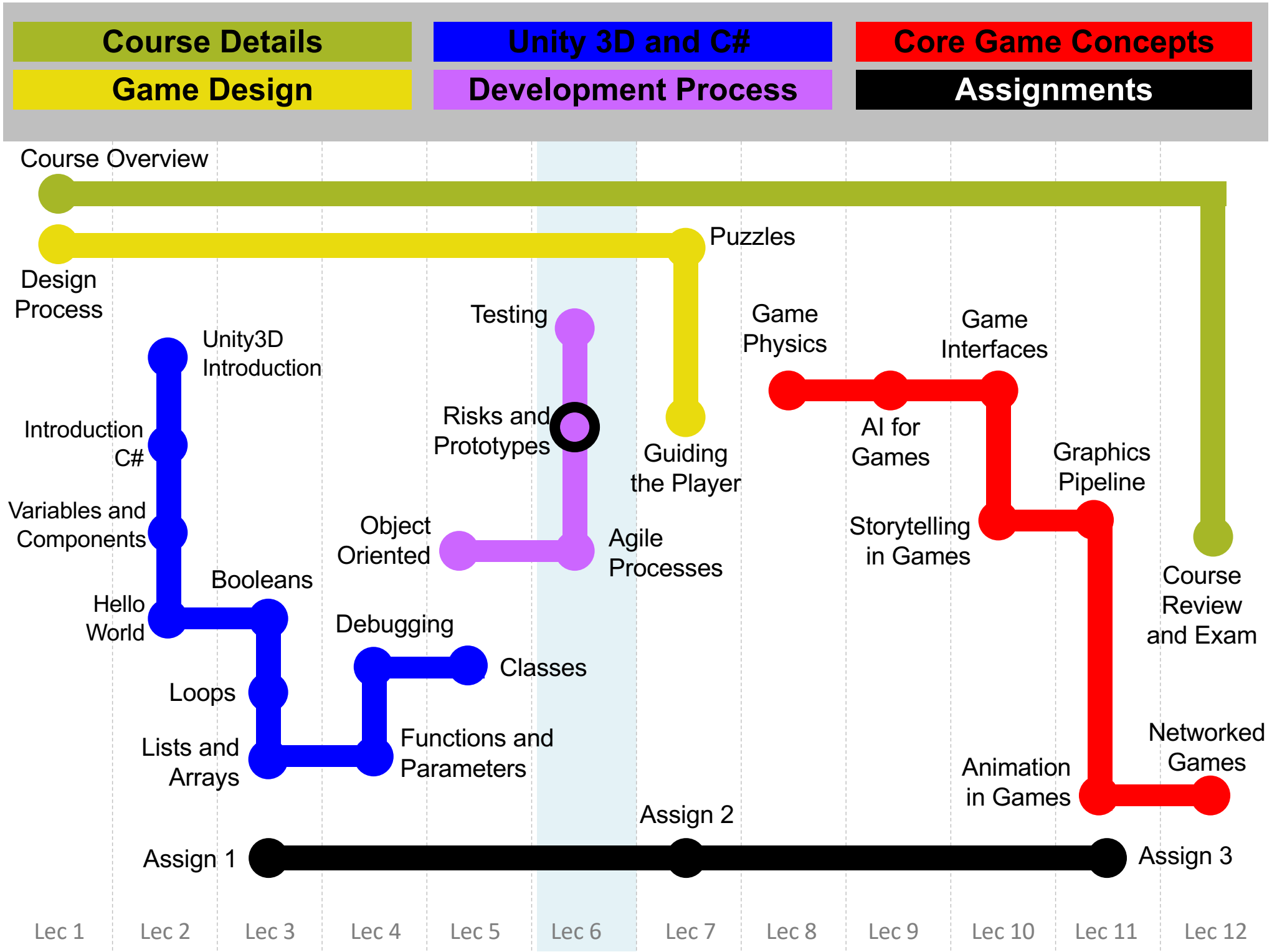
Week 06

Module 6.2

Risks and Prototypes

Course Overview

Lec	Start Week	Modules	Topics	Assignments
1	3 Aug	Mod 1.1, 1.2	Course Overview, Design Process	
2	10 Aug	Mod 2.1, 2.2, 2.3, 2.4	Unity3D Introduction, Introduction C#, Variables and Components, Hello World	
3	17 Aug	Mod 3.1, 3.2, 3.3	Booleans, Loops, Lists and Arrays	Assign 1 21 Aug, 11:00 pm
4	24 Aug	Mod 4.1, 4.2	Functions and Parameters, Debugging	
5	31 Aug	Mod 5.1, 5.2	Classes, Object Oriented	
6	7 Sep	Mod 6.1, 6.2, 6.3	Agile Processes, Risks and Prototypes, Testing	
7	14 Sep	Mod 7.1, 7.2	Puzzles, Guiding the Player	Assign 2 18 Sep, 11:00 pm
8	21 Sep	Mod 8.1	Game Physics	
9	12 Sep	Mod 9.1	AI for Games	
10	19 Oct	Mod 10.1, 10.2	Game Interface, Storytelling in Games	
11	26 Oct	Mod 11.1, 11.2	Graphics Pipeline, Animation in Games	Assign 3 1 Nov, 11:00pm
12	2 Nov	Mod 12.1, 12.2	Networked Games, Course Review	


Course Details
Unity 3D and C#
Core Game Concepts
Game Design
Development Process
Assignments
Course Overview
Puzzles
Design Process
Testing
Game Physics
Game Interfaces
Unity3D Introduction
Risks and Prototypes
Guiding the Player
AI for Games
Graphics Pipeline
Introduction C#
Variables and Components
Object Oriented
Agile Processes
Storytelling in Games
Course Review and Exam
Booleans
Hello World
Debugging
Classes
Loops
Functions and Parameters
Lists and Arrays
Networked Games
Animation in Games
Assign 2
Assign 1
Assign 3
Lec 1
Lec 2
Lec 3
Lec 4
Lec 5
Lec 6
Lec 7
Lec 8
Lec 9
Lec 10
Lec 11
Lec 12

Risks and Prototypes – Topics

- Development Cycle
- Why Project Fail
- Risk Management
- Development Team
- Development Processes
- Risk Management
- Prototyping
- Development Environments / Tools

Games and Software

Like all software development games require:

- good project management
- good requirements analysis
- good risk assessment
- a good understanding of the development process

Game Development Life Cycle

Concept - Create game concepts

Preproduction - Flesh out concepts, often with prototypes and vertical slices

Production - Build the game to first playable

Postproduction - Final QA, Acceptance testing, localization, ratings, box art and details, marketing,

Aftermarket - Support, promotions, patches, post-mortems

Why do projects fail?

The traditional problems associated with software development

- cost and time overruns
- failure to meet requirements

have also been a problem for game developers.

Why do projects fail?

The traditional problems associated with software development

- cost and time overruns
- failure to meet requirements

have also been a problem for game developers.

Because a game project also requires not just software but also a number of artistic resources and the interaction of cross-discipline team members the project management is even more critical.

Project Risks

We will examine the typical risks associated with all software projects.

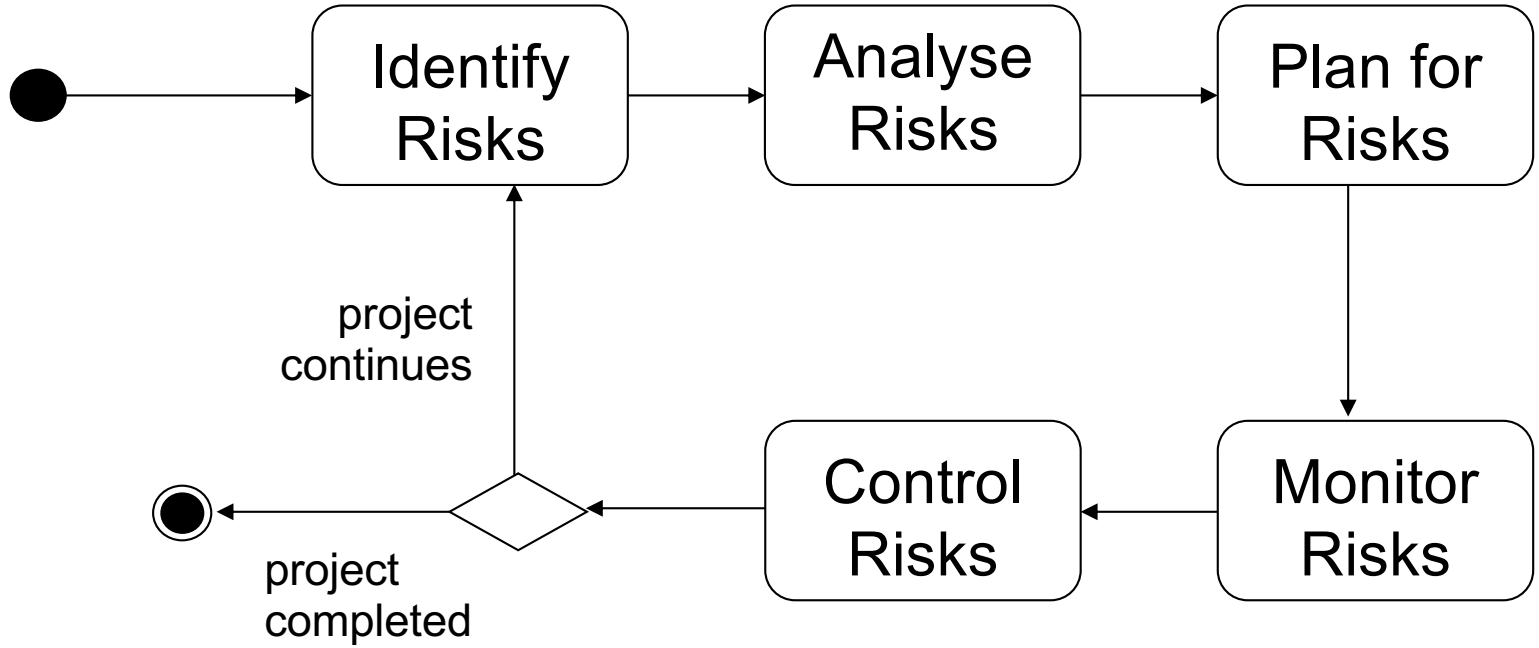
Most of these risks will also be an issue in game projects (including your own).

The risks need to be understood and managed if you wish to successfully produce a computer game

Managing Project Risks

Risks need to be constantly managed in a project.

They need to be identified, analysed, mitigation strategies planned, monitored and controlled.



Software Failure

Some reasons why software projects fail *(avoid these)*

Development Phase	Failure Symptom
Requirements Analysis	No written requirements Incompletely specified requirements No user interface mock-up No end-user involvement
Design Specification	Lack of or insufficient design documents Poorly specified data structures and file formats Infrequent or no design reviews
Implementation	Lack of or insufficient coding standards Infrequent or no code reviews Poor in-line code documentation
Unit Test & Integration	Insufficient module testing Lack of proper or complete test site Lack of independent quality assurance group
Beta Test Release	Complete lack of beta test Insufficient duration of beta test insufficient number of beta testers Wrong beta testers selected
Maintenance	Too many bug reports

Hair, B 1999, 'Risk Reduction in Software Development', viewed 22 August 2007, <http://http://www.cashandassociates.com/riskpaper.doc>

Software Success

Some factors in ensuring software projects succeed *(encourage these)*

Factor	Rank	%
Clear Statement of Requirements	1	7.30
Proper Planning	2	7.09
Clear Vision & Objectives	3	7.03
Hard Working, Focused Staff	4	6.86
Competent Staff	5	6.77
User Involvement	6	6.53
Realistic Expectations	7	6.44
Executive Management Support	9	6.24
Ownership	11	5.53
Smaller Project Milestones	12	5.26
Other		35.13

Hair, B 1999, 'Risk Reduction in Software Development', viewed 22 August 2007, <http://http://www.cashandassociates.com/riskpaper.doc>

Project Challenges

Some factors that challenge the success of software projects succeed *(look out for these)*

Factor	Rank	%
Incomplete Requirements & Specifications	1	7.66
Changing Requirements & Specifications	2	7.51
Unclear Objectives	2	7.51
Unrealistic Expectations	4	7.04
Lack of User Involvement	5	6.77
Lack of Executive Support	7	6.42
Lack of Resources	9	5.56
Technology Incompetence	10	5.21
New Technology	10	5.21
Unrealistic Time Frame	12	5.02
Other		36.09

Hair, B 1999, 'Risk Reduction in Software Development', viewed 22 August 2007, <http://http://www.cashandassociates.com/riskpaper.doc>

Managing your Risks

Control negative (bad) risks

Avoidance	Eliminate a risk
Acceptance	Accept consequences (if low impact)
Transference	Shift consequence & responsibility to third party
Mitigation	Reduce impact or probability of risk

Pragmatic Software Newsletters 2003, 'Tips for Managing Risk in Software Projects', viewed 22 August 2007, <http://www.pragmaticsw.com/Newsletters/newsletter_2003_08_SP.htm>

Managing your Risks

Control negative (bad) risks

Avoidance	Eliminate a risk
Acceptance	Accept consequences (if low impact)
Transference	Shift consequence & responsibility to third party
Mitigation	Reduce impact or probability of risk

Control positive (good) risks

Exploitation	Be proactive to make the risk (good thing) happen
Acceptance	Accept consequences (if low impact)
Sharing	Share ownership with another party (synergy)
Enhancement	Increase positive outcomes of risk

Pragmatic Software Newsletters 2003, 'Tips for Managing Risk in Software Projects', viewed 22 August 2007, <http://www.pragmaticsw.com/Newsletters/newsletter_2003_08_SP.htm>

Game Project Success

Some tips for success of your game project

(these are adapted from lessons learned in software projects)

1. User (Player) Involvement
2. Clear Statement of Requirements
3. Proper Planning
4. Realistic Expectations
5. Smaller Project Milestones
6. Ownership
7. Clear Vision & Objectives
8. Hard-Working, Focused Staff

CHAOS Report 1994, 'The CHAOS Report', viewed 22 August 2007, <http://www1.standishgroup.com/sample_research/chaos_1994_1.php>

Development Team

- Producers
- Programmers
- Artists
- Animators
- Designers
- Audio
- QA

Team - Producers

- Producers
- Programmers
- Artists
- Animators
- Designers
- Audio
- QA

Project management – oversee the development of the game

Work with the team to set/track goals and milestones

Liaise with stakeholders (e.g. marketing, license holders, localization, executive management, office management)

Team - Programmers

- Producers
- Programmers
- Artists
- Animators
- Designers
- Audio
- QA

Engine

Work on rendering and performance
Often very senior

Gameplay

Create AI, UI, other gameplay
Work closely with designers

Tools

Tools used by artists and designers

Team – Artists/Animators

Producers

Programmers

Artists

Animators

Designers

Audio

QA

- Concept
- Modelling
- Rigging
- Environment
- Character
- Effects
- Animate Characters

Team - Designers

- Producers
- Programmers
- Artists
- Animators
- Designers
- Audio
- QA

Game designers set the high level game rules of the game

Level designers whitebox and build individual levels

Represent the player

Team - Audio

- Producers
- Programmers
- Artists
- Animators
- Designers
- Audio
- QA

Foley (everyday sounds) - the sound effects used as part of the game
(eg. Foodsteps, gun sounds)

Music (narrative, emotion)

User Interface (enhance usability)

Informative (enhance performance)

Team – Quality Assurance

- Producers
- Programmers
- Artists
- Animators
- Designers
- Audio
- QA

Test the game

A game might have multiple QA teams for different stages of the game

Embedded QA

Team QA

First Party approval

Localisation (country, culture,..)

Development Processes

A software development process defines a series of steps or phases that are used to develop software.

A number of different processes have been used and we will look at a few different well-known processes.

(We have already looked at Agile processes like Scrum)

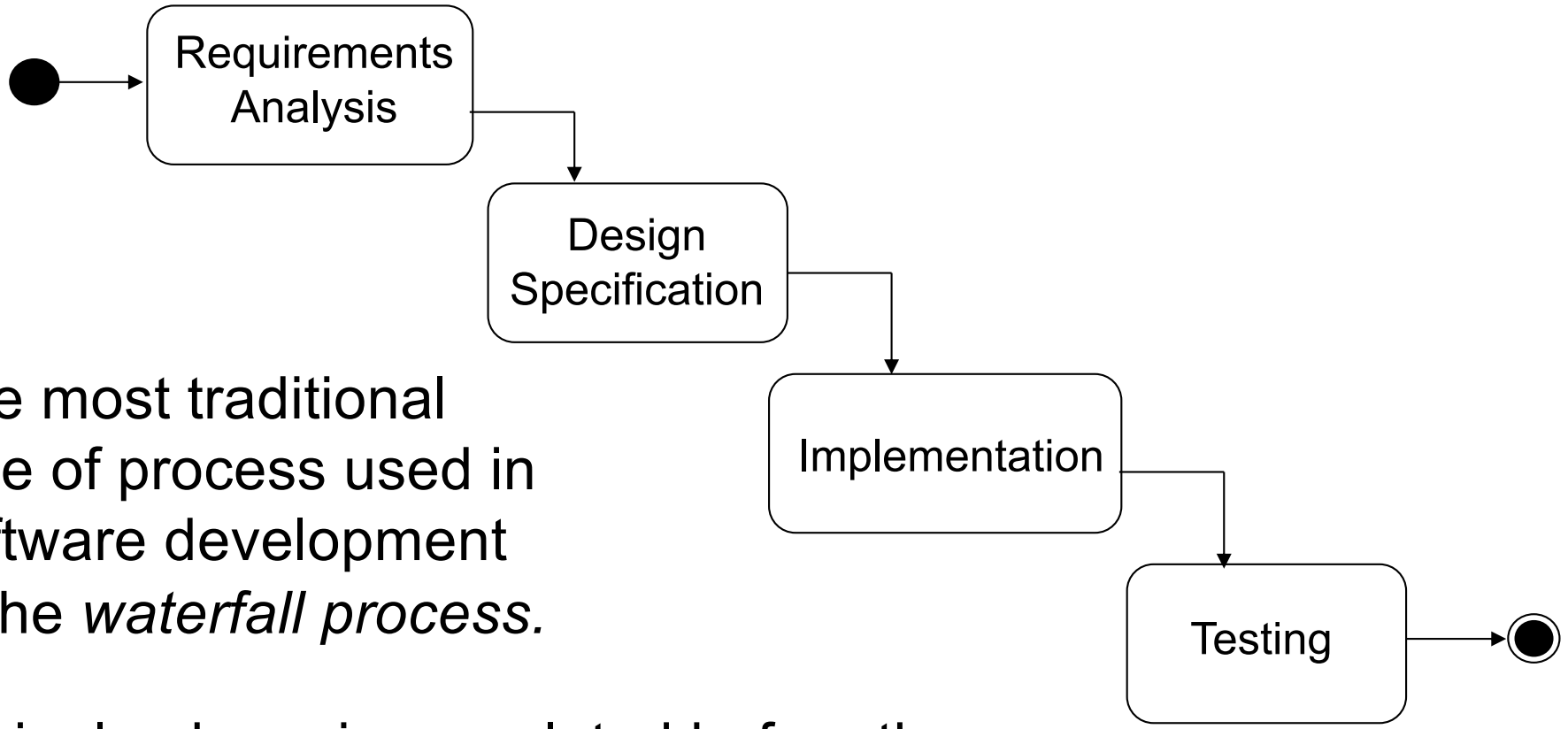
We will also discuss how they are used in game development.

Development Processes

Different processes might be more appropriate for different size projects. The best process will depend somewhat on the nature of the project.

- A small independent game development will typically involve a small amount of code and artwork.
- A large highly-produced, cross platform blockbuster will involve a lot more resources, staff and software.

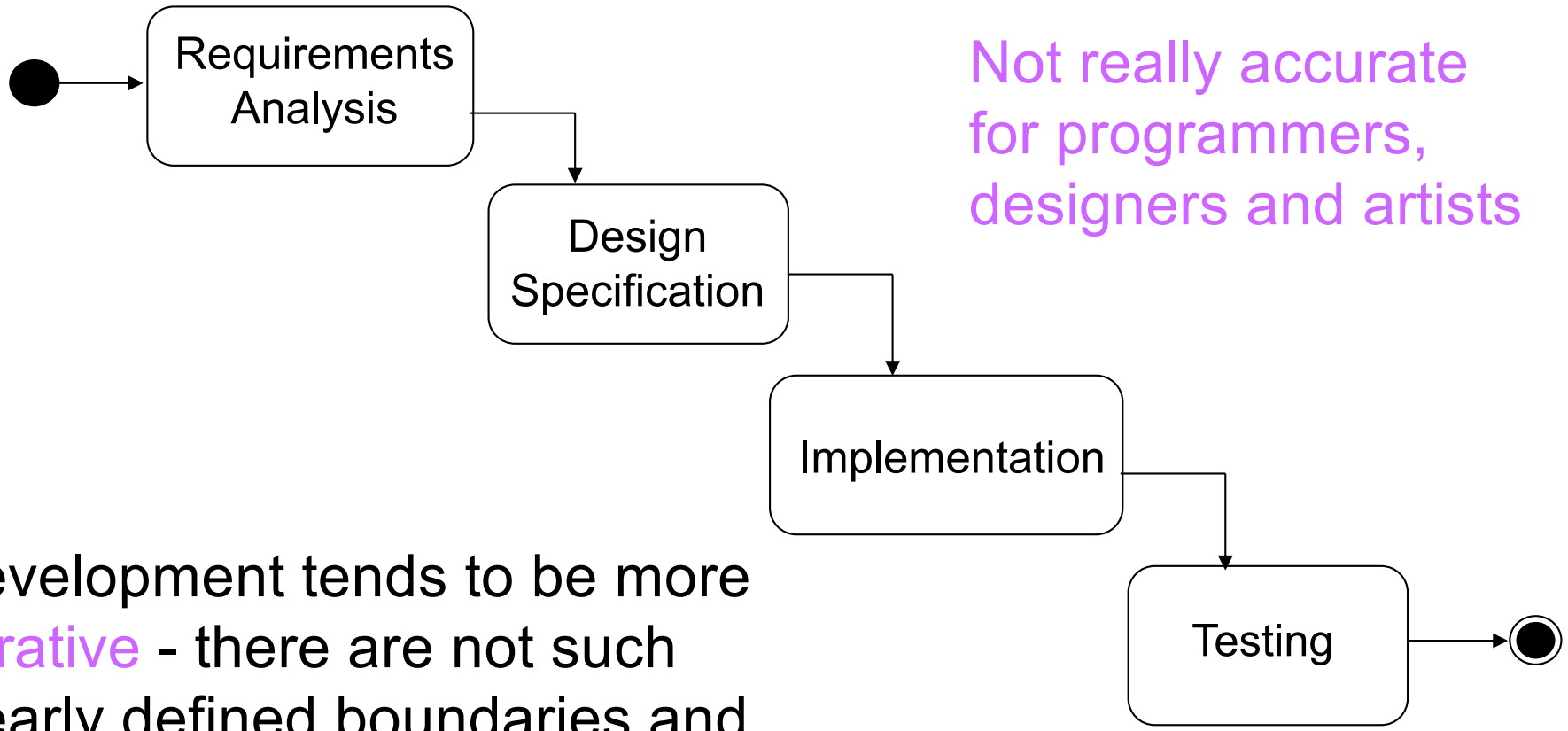
Waterfall Processes



The most traditional type of process used in software development is the *waterfall process.*

A single phase is completed before the next phase begins. Each phase has clearly defined deliverables.

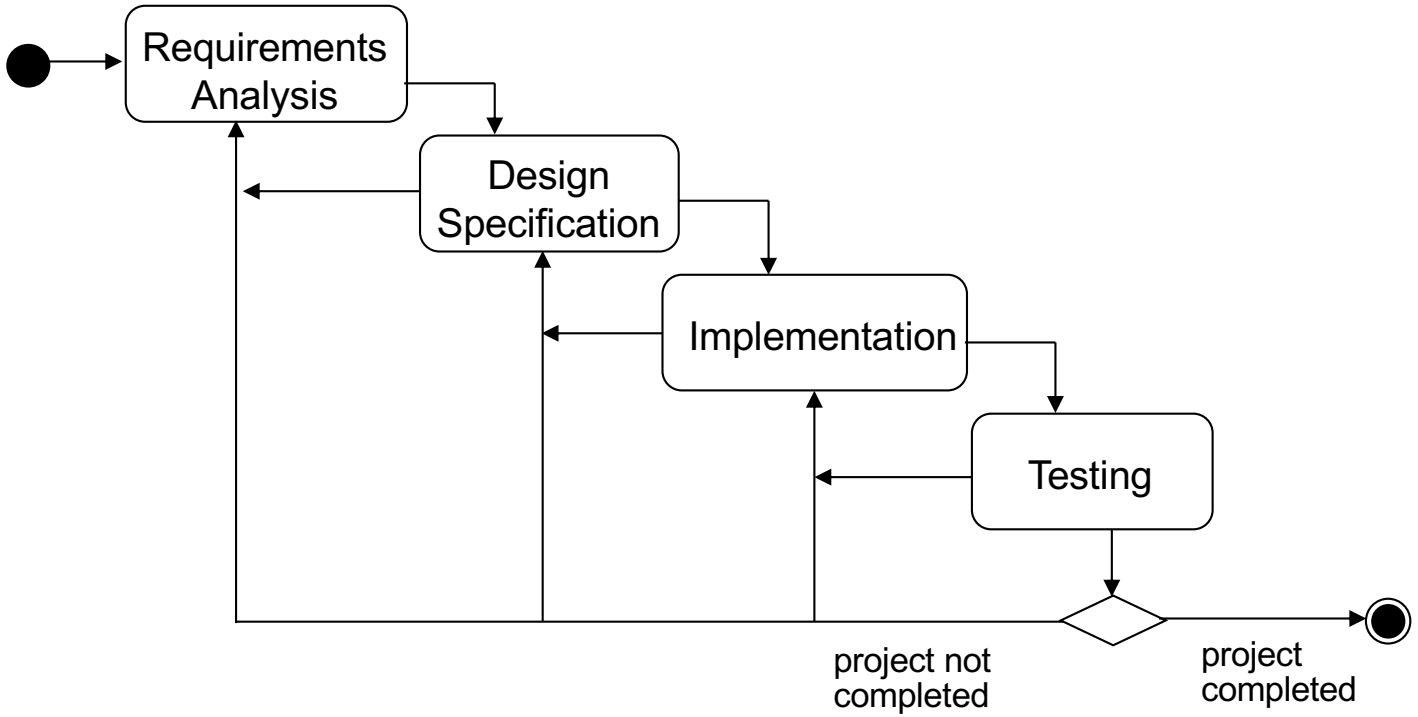
Waterfall Processes



Not really accurate for programmers, designers and artists

Development tends to be more iterative - there are not such clearly defined boundaries and each of the phases may have to be revisited many times during the project's lifetime.

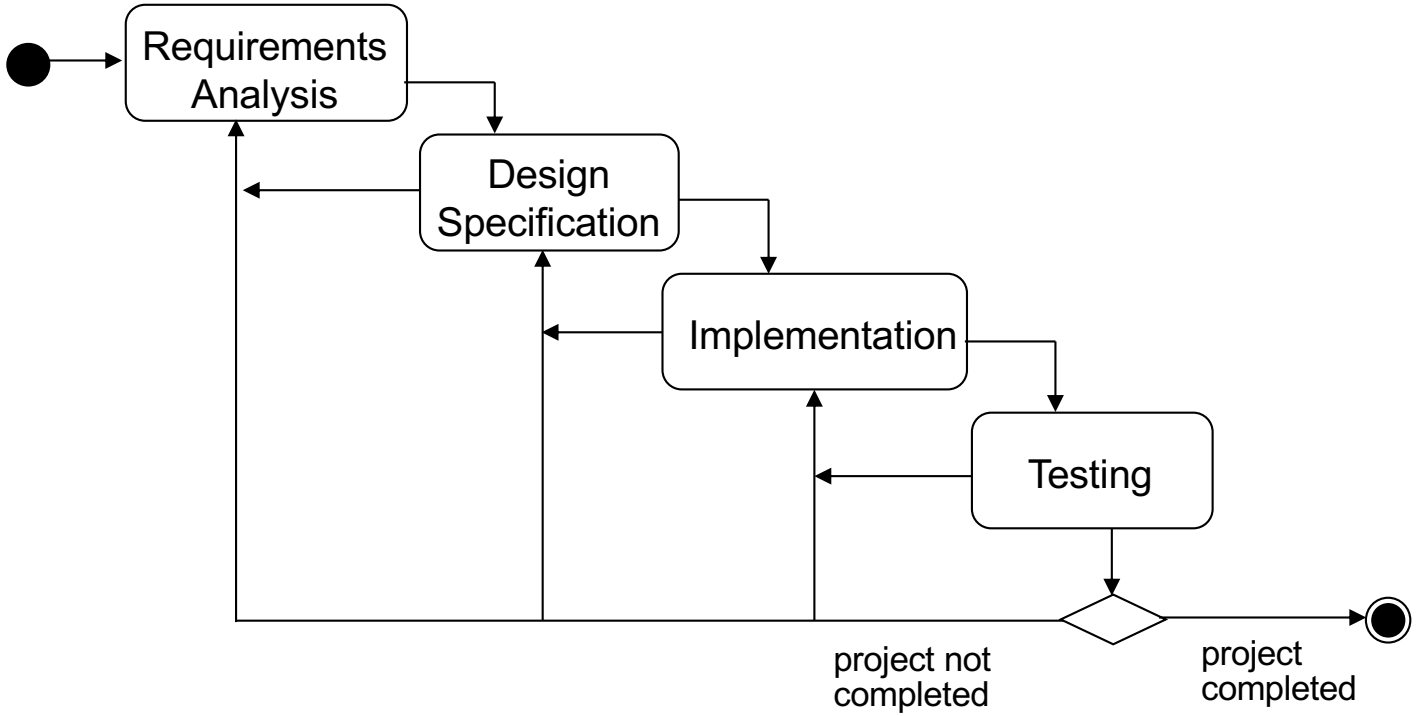
Iterative Processes



In these iterative processes - the different phases may need to be revisited many times.

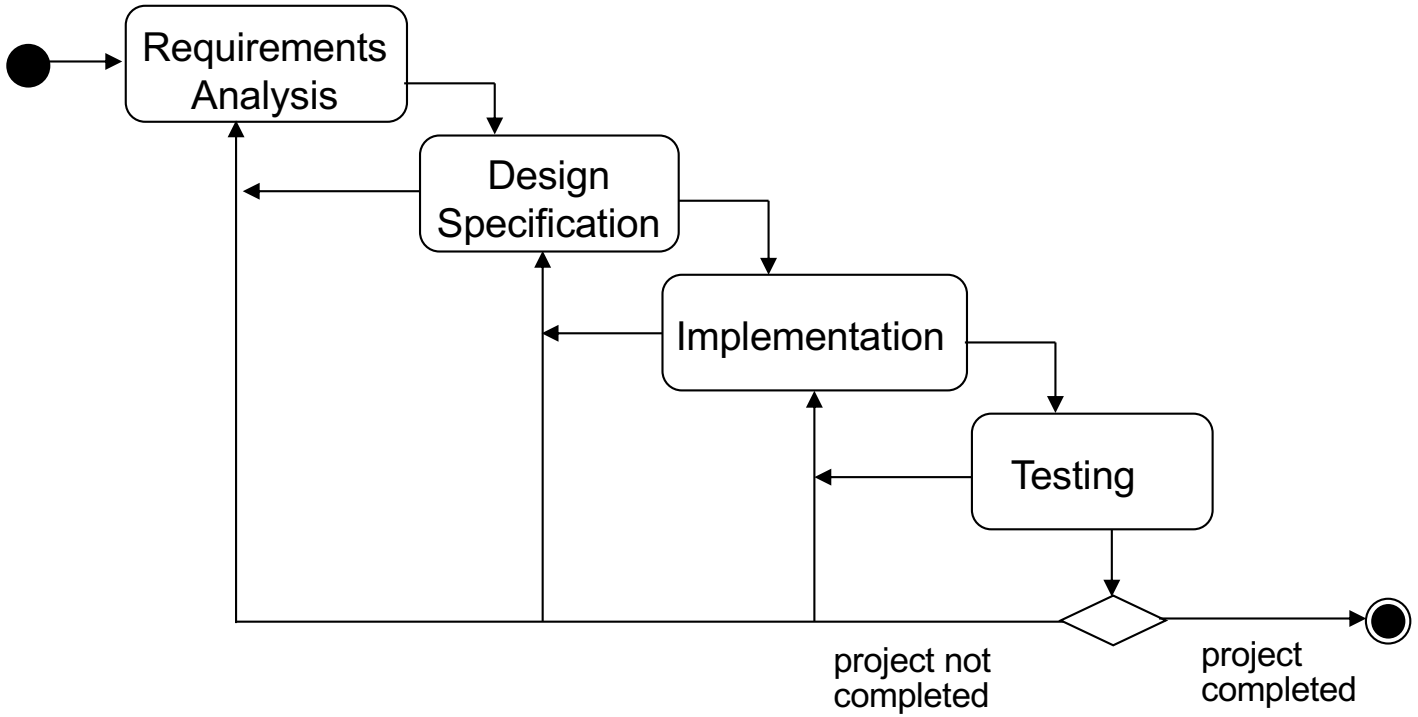
- For example, after implementation begins it may be necessary to return to the design or requirements phase.
- Testing will often reveal flaws in implementation and perhaps design.

Iterative Processes



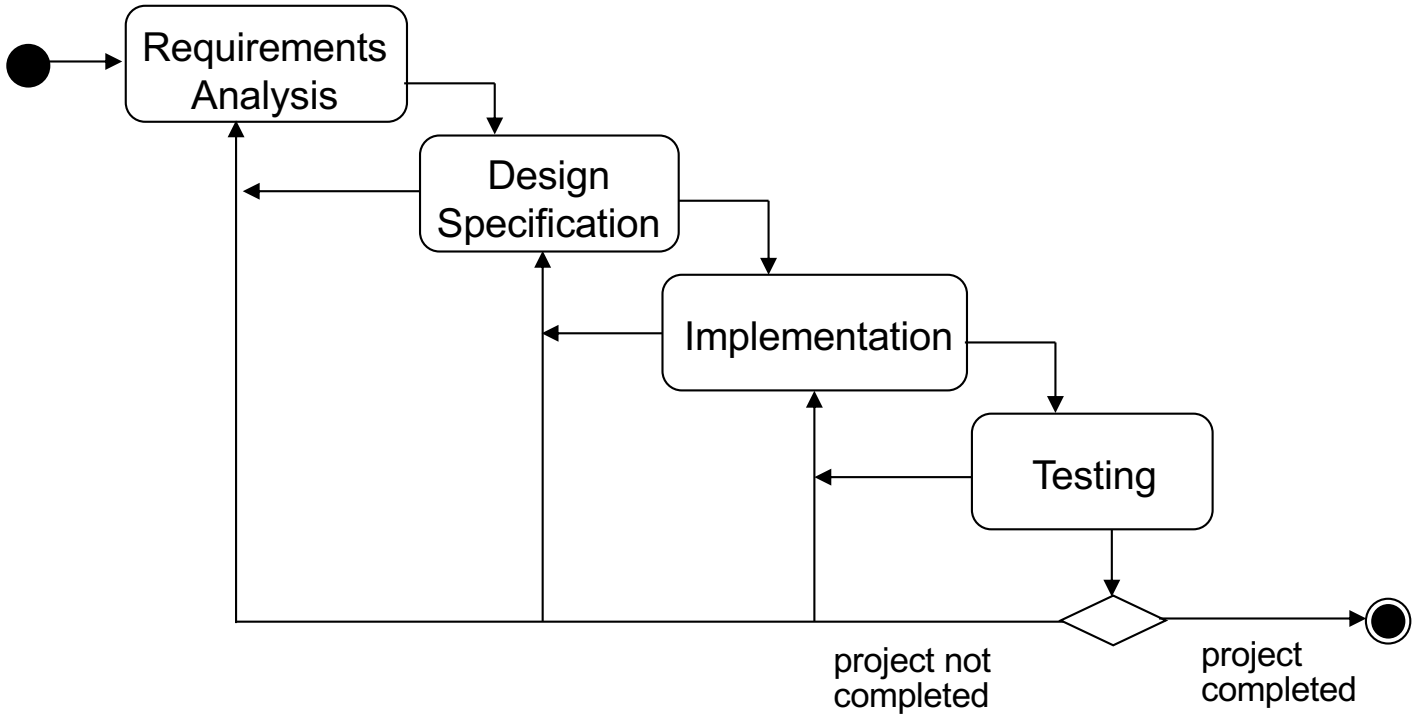
These iterative processes tend to be a better match to the reality experienced by programmers - they better define the typical workflow on a project.

Iterative Processes



It is also important to recognise that changes that occur to requirements and design typically become much more expensive (in time and resources) to change as the project matures. e.g. a design flaw found in the testing phase may take up to 100-1000 times more effort to fix.

Iterative Processes

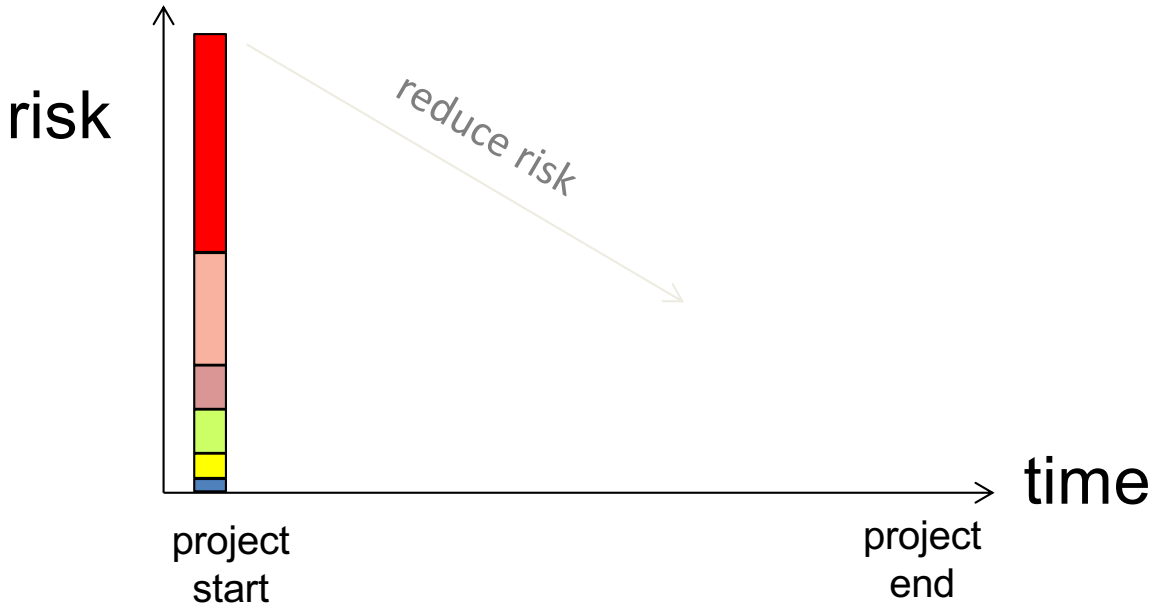


There are a number of different iterative processes described and they have a number of useful features

Two of these typical features are "risk management" and "prototyping".

Risk Management

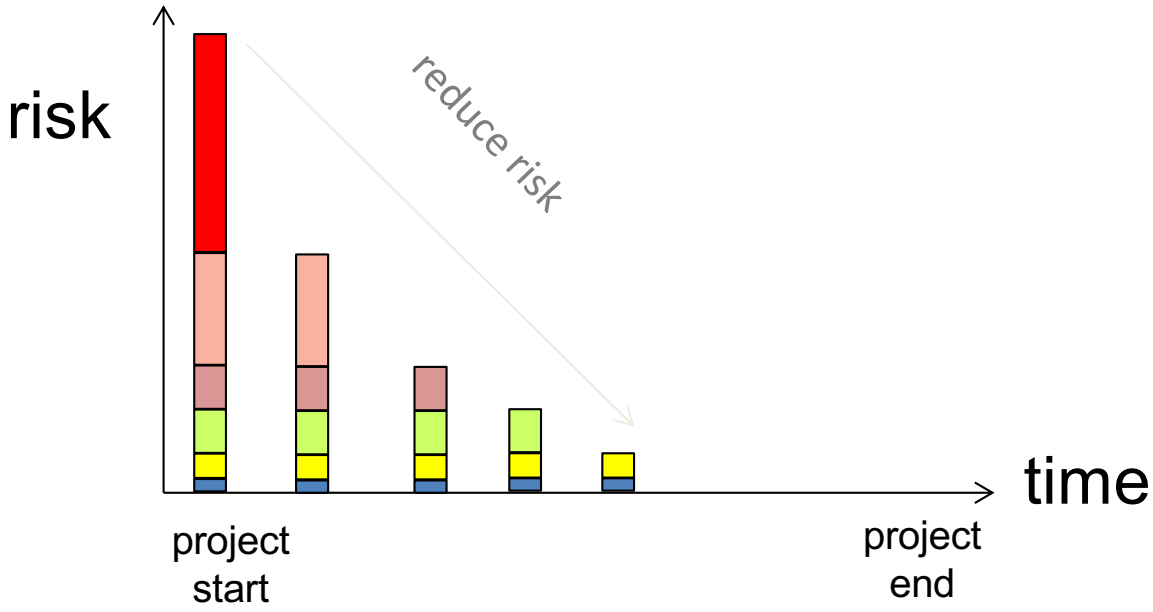
The key with introducing risk management into an iterative process is to make sure that the risk of project failure reduces as the project proceeds.



Risk Management

This means addressing the highest risk elements of the project as soon as possible.

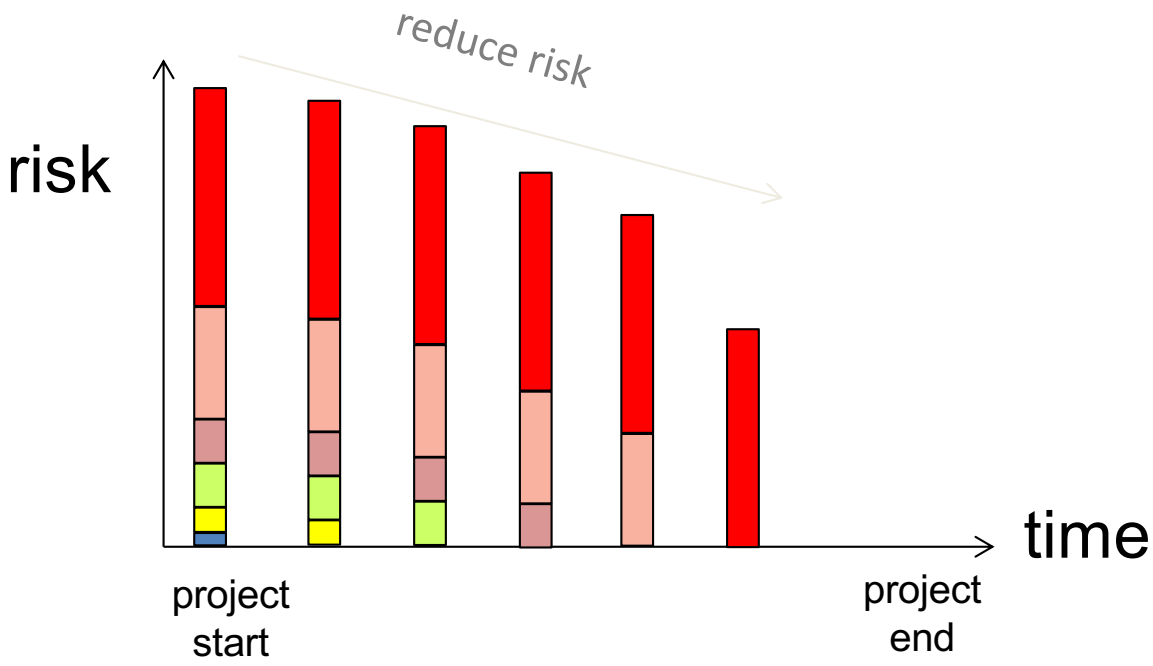
If some possible problem might cause the project to fail then it should be solved first. It is not sensible to leave it to later.



Risk Management

This means addressing the highest risk elements of the project as soon as possible.

Not This!



Prototyping

Many iterative processes also proceed by producing a number of prototypes. Prototypes are early versions of the software which are not fully functional.

Prototyping

Many iterative processes also proceed by producing a number of prototypes. Prototypes are early versions of the software which are not fully functional.

Prototyping is very useful for resolving uncertain requirements or features. It is good for early testing (e.g. checking the gameplay). It is also a good way to address, up front, the key risks in the project.

Prototyping

Many iterative processes also proceed by producing a number of prototypes. Prototypes are early versions of the software which are not fully functional.

Prototyping is very useful for resolving uncertain requirements or features. It is good for early testing (e.g. checking the gameplay). It is also a good way to address, up front, the key risks in the project.

A prototype may be thrown away or it may be evolved into the the final software. We will be working with some prototypes in this course.

Agile Processes

There is also another group of processes, called Agile processes that have traditionally been associated with game development.

(Though this is more true of smaller independent game development than larger game projects)

They are usually light weight processes they focus very much on short term problems and on delivering outcomes.

Agile Processes

They use an evolutionary model of development so that the project team responds quickly to each successive problem that arises.

The software (game) "evolves" as each short term challenge is met.

Of course your projects are small independent games so this process description may also be relevant.

Agile Processes

Agile methods (such as SCRUM) share a lot in common with other processes we have discussed:

- Iterative - repeated, but very short phases
- Promotes evolutionary change during life-cycle
- Focuses on real-time communication rather than documentation
- More typical of what happens (from a programmer's perspective)
- Requires experience to manage (from a manager's perspective)
- But there are still clearly defined tasks and outcomes from each phase - but these are reviewed frequently (daily or weekly)

SCRUM 2007, 'SCRUM (Development)', viewed 23 August 2007, <http://en.wikipedia.org/wiki/Scrum_%28development%29>

SCRUM Features

A backlog of prioritized work items is maintained;

The aim is to complete a fixed set of work items in a series of short iterations (or sprints);

Frequent intermediate deliveries with working functionality.

A brief daily meeting (or scrum) is held - where progress is explained, upcoming work described, impediments are raised.

- What have you done since yesterday?
- What are you planning to do by tomorrow?
- Do you have any problems preventing you from accomplishing your goal?

Frequent risk and mitigation plans are developed by the development team itself.

Development Environment / Tools

- Version control
- Bug tracking
- Development Kits
- IDEs
- Art software
- Audio software
- Project Management Software