

# Effective Software Development in the 21st Century

## *The New Face Of Software Engineering*

**Alistair Cockburn**  
<http://Alistair.Cockburn.us>

Slide 1

©Alistair Cockburn 2008



*Developing software consists of people making  
ideas concrete in an economic context*

People inventing and communicating,

Solving a problem

*they don't yet understand*

which keeps changing

Creating a solution

*they don't yet understand*

which keeps changing

Expressing ideas in languages *they don't understand*

which keep changing

To an interpreter unforgiving of error

Making decisions with limited resources

where every choice has economic consequences

Slide 2

©Alistair Cockburn 2008



## Foundation 1

Software development is a *craft*



## PEOPLE learn *skills* in 3 stages



Shu: *Follow*

Learn a technique



Ha: *Break away*

Collect techniques



Ri: *Fluent*

Blend techniques



## *Craft* is continuously growing skills in a medium

- 1 Deciding what to build
- 2 Managing (*people and projects*)
- 3 Modeling
- 4 Designing the external view
- 5 Large-scale design (*architecting*)
- 6 Fine-scale design (*programming*)
- 7 Validating the work

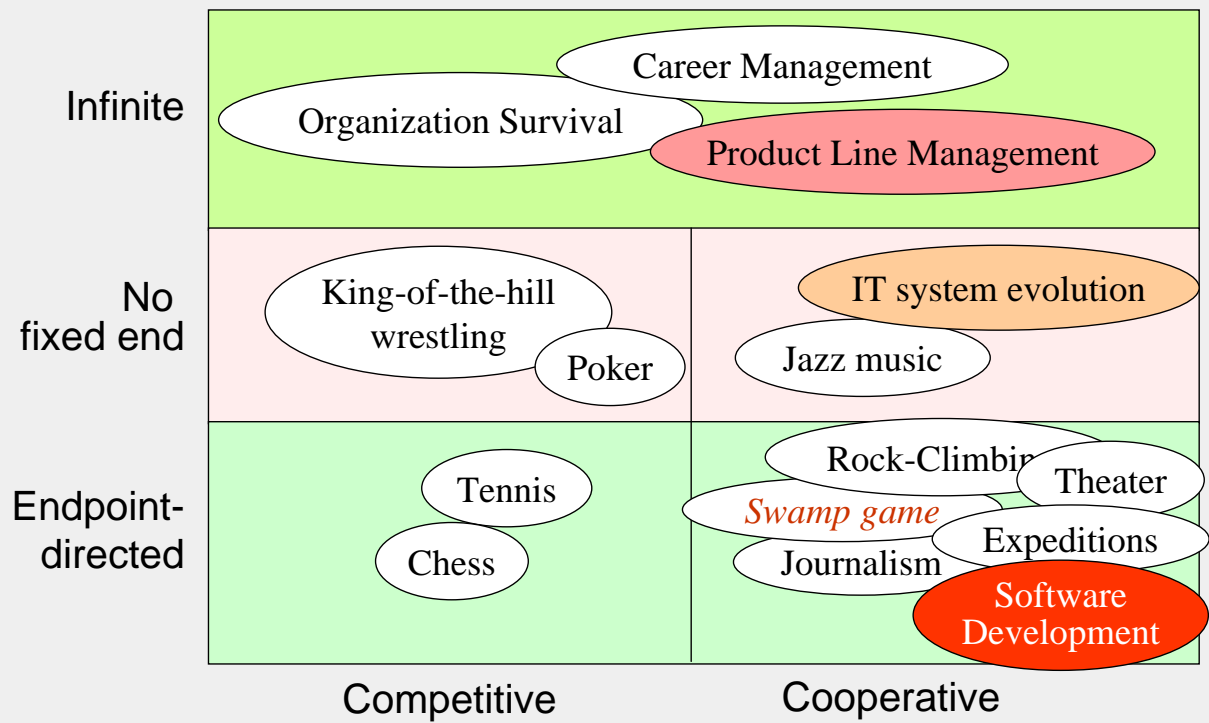


## Foundation 2

Software development is a  
**Cooperative Game**  
of invention and communication



Games have moves, strategies & don't repeat!



Each project has conflicting subgoals:

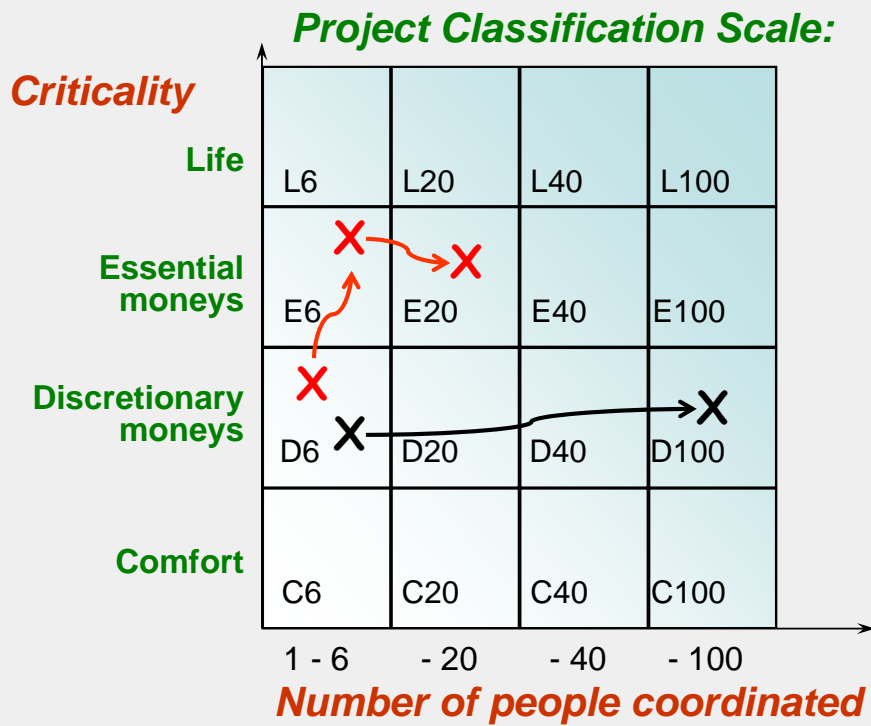
**Deliver** this software

and

**Set up** for the next game!



## Adjust to the situation

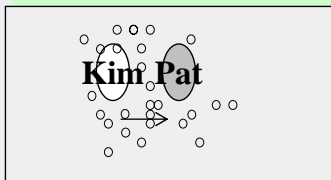


Slide 9

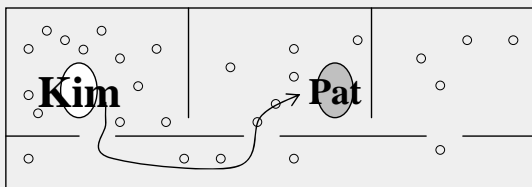
©Alistair Cockburn 2008



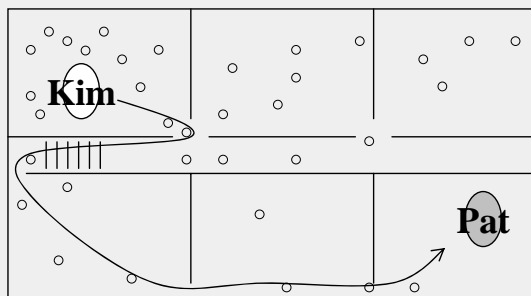
## Distance is *expensive*



Programming in pairs



**12 people:**  
= \$100,000 / yr penalty



**12 people**  
= \$300,000 / yr penalty

“Distance Matters”  
“Managing the Flow of Technology”

Slide 10

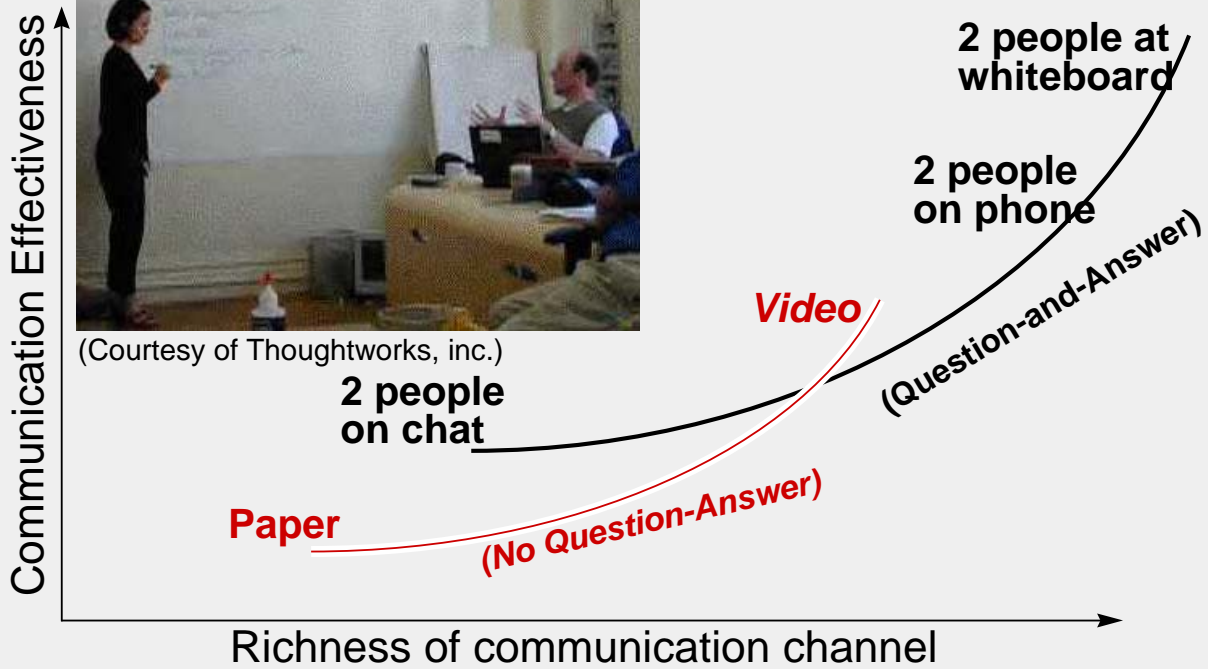
©Alistair Cockburn 2008



*Face-to-face is the most effective -  
Consider video !*

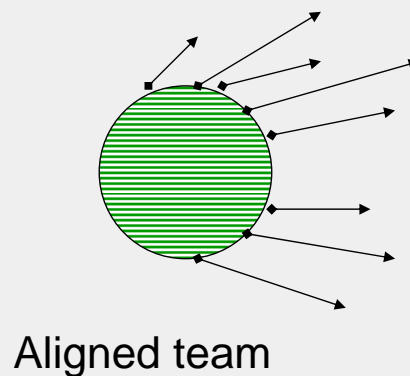
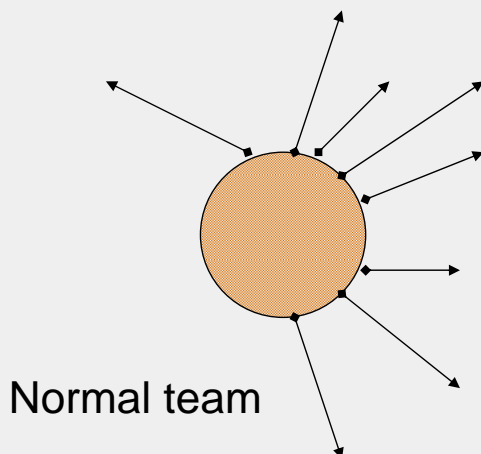


(Courtesy of Thoughtworks, inc.)



**Improve amicability ; align people's goals**

*Amicability* = willingness to listen with good intention



## *People* issues determine a project's speed

Can they easily detect something needs attention?

(Good at Looking Around)

Will they care enough to do something about it?

(Pride-in-work; Amicability)

Can they effectively pass along the information?

(Proximity; face-to-face)

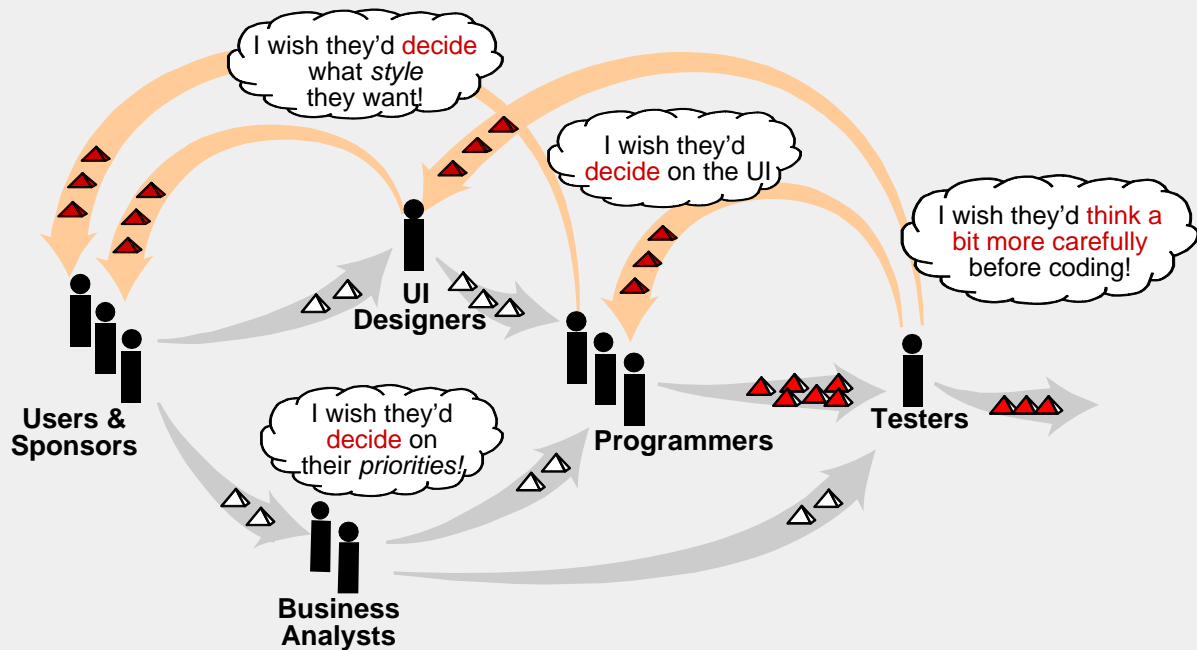


## Foundation 3

Use *Lean* processes



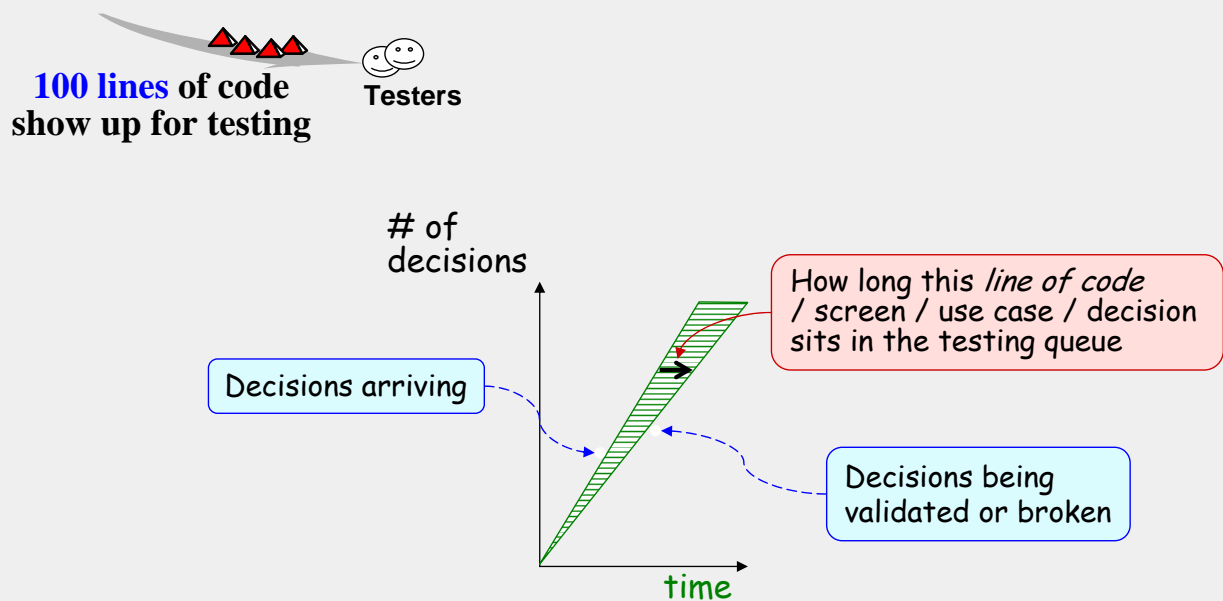
## Unvalidated decisions are design inventory



Slide 15

©Alistair Cockburn 2008

## Lean Manufacturing strategies apply continuous flow

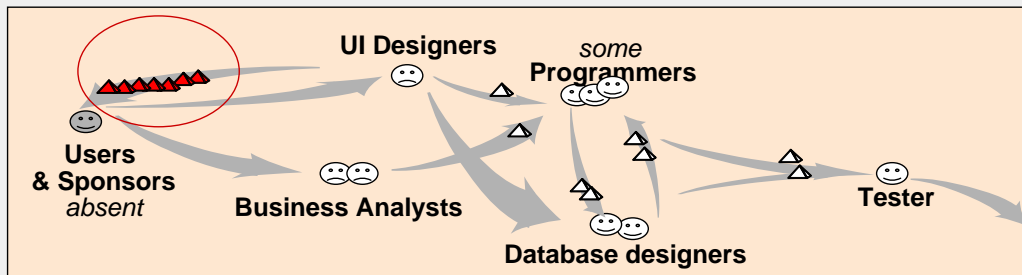
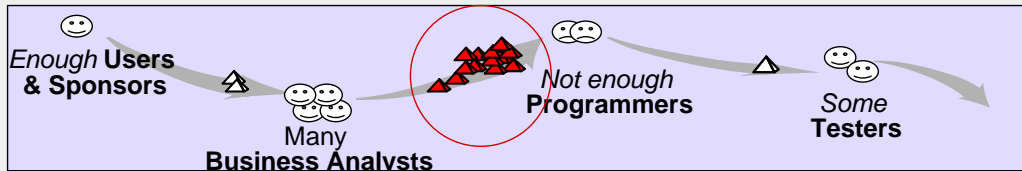
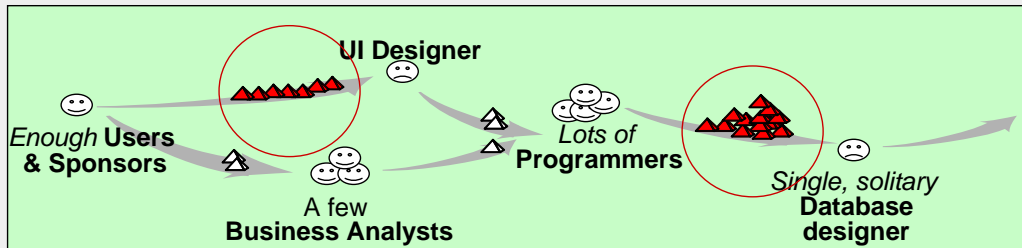


Slide 16

©Alistair Cockburn 2008



## Lean Manufacturing strategies apply attend to backed-up queues



Slide 17

©Alistair Cockburn 2008



## Foundation 4

Consider Design as  
*Knowledge Acquisition*

Slide 18

©Alistair Cockburn 2008



We best how to build the system *after* we ship it!

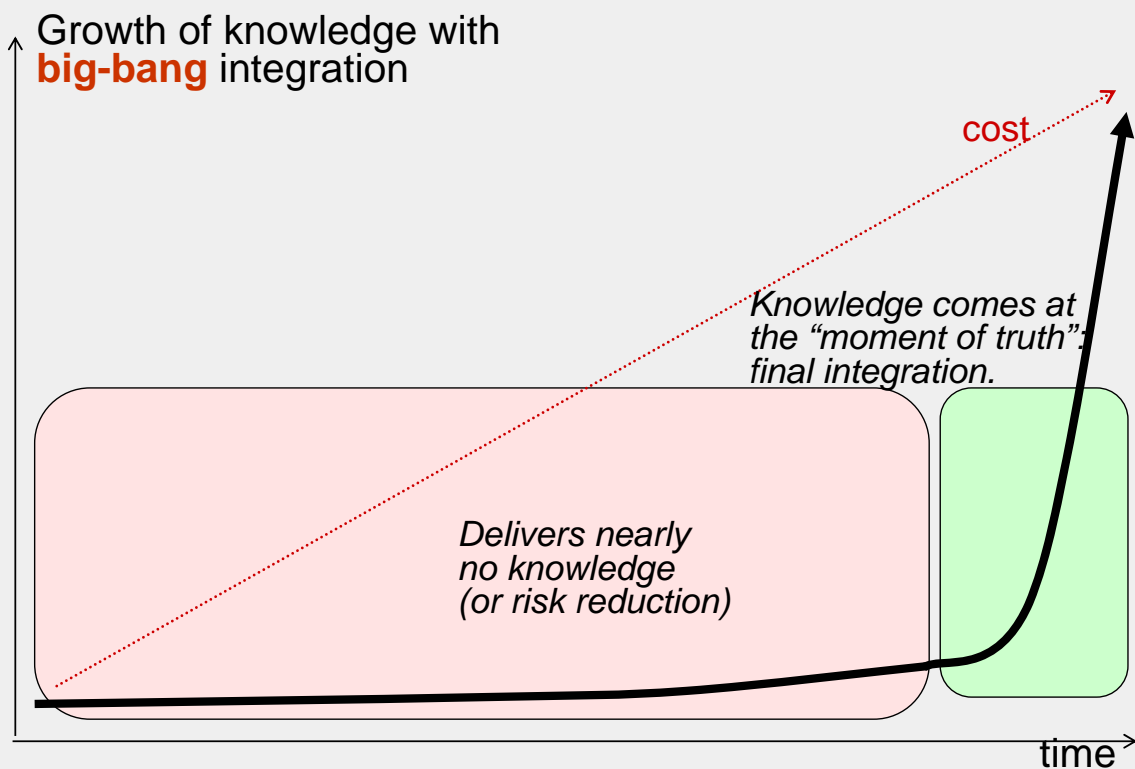
*How can we make use of this depressing fact?*

Slide 19

©Alistair Cockburn 2008



Waterfall is a *late-learning* strategy



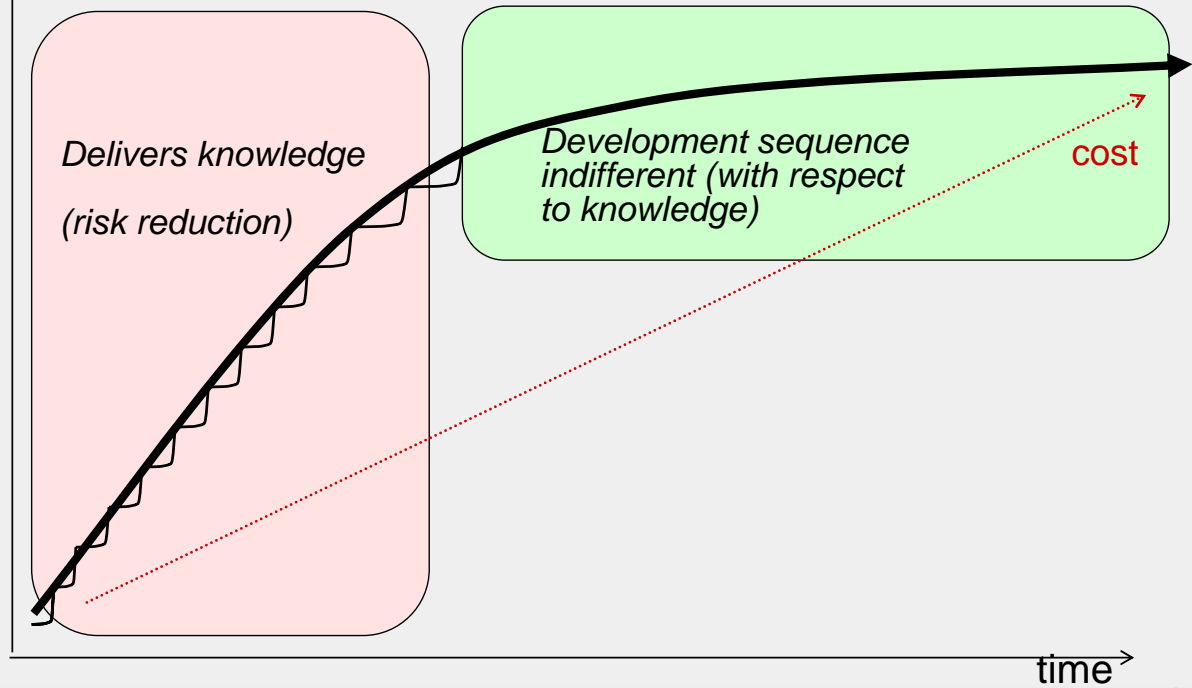
Slide 20

©Alistair Cockburn 2008



## We can pay to *learn* early in the project

Growth of knowledge with  
*early, continuous* integration

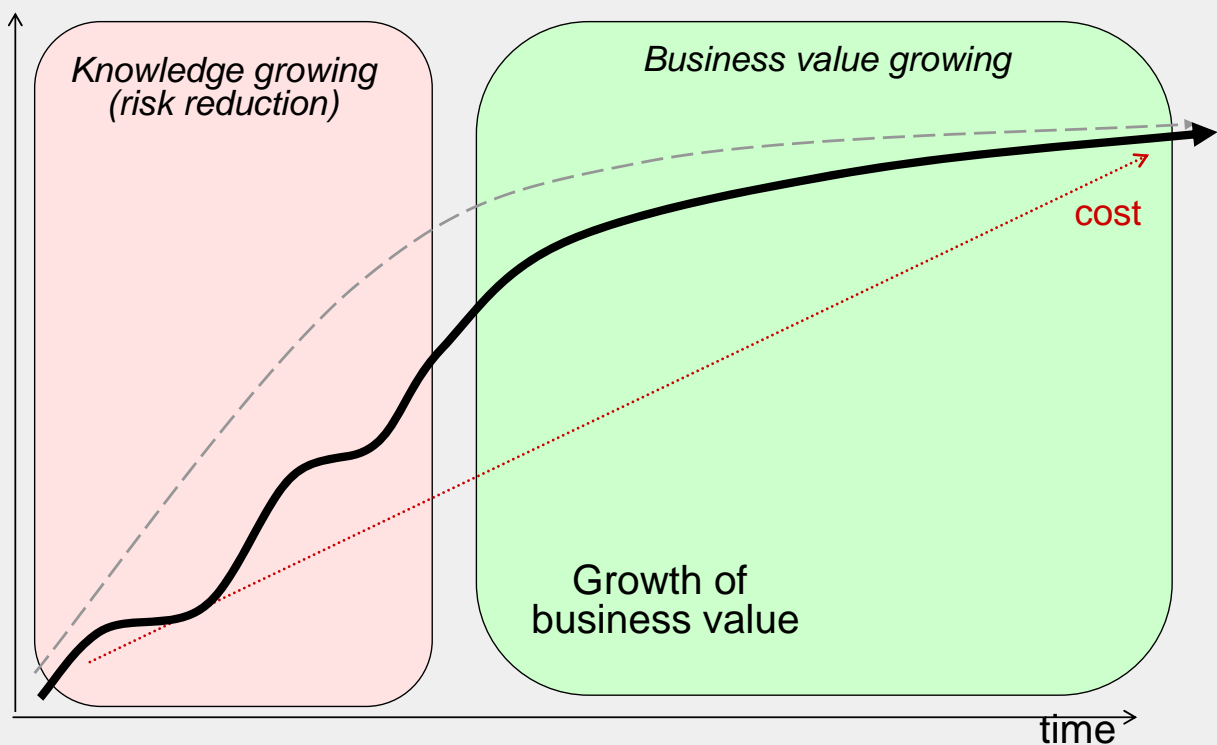


Slide 21

©Alistair Cockburn 2008



## Develop for *business value* once risks are down

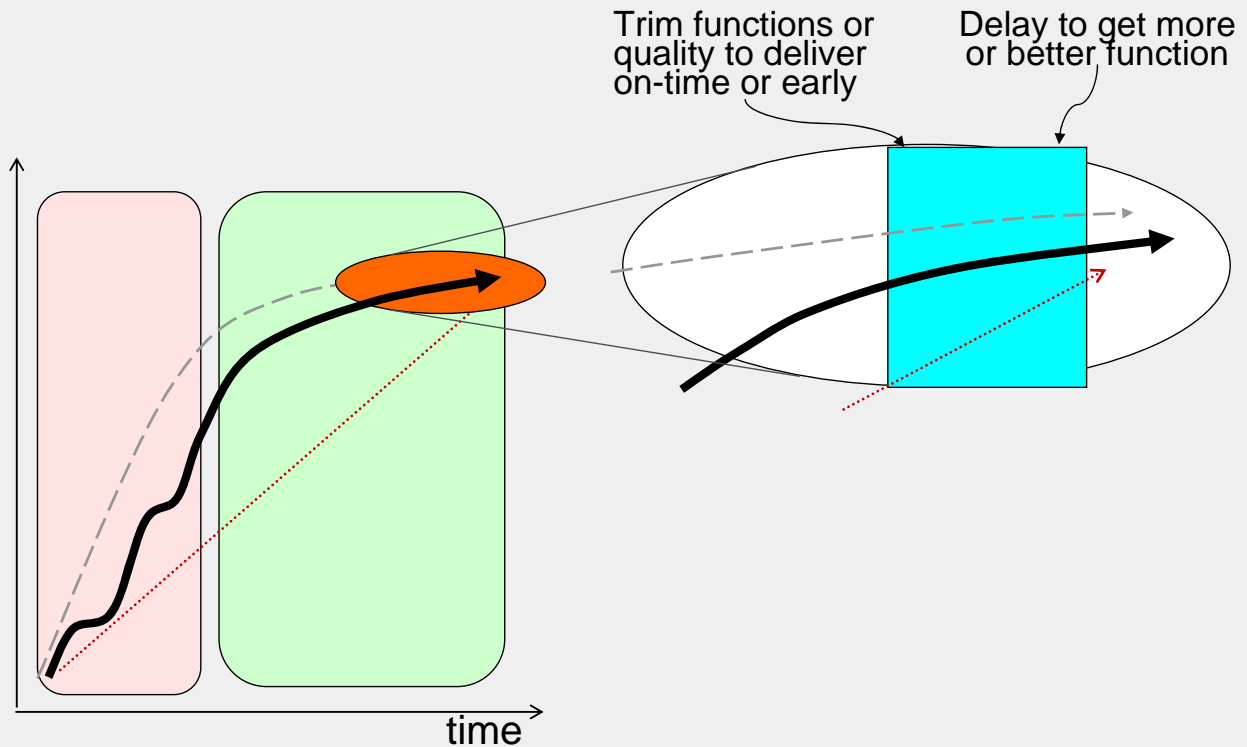


Slide 22

©Alistair Cockburn 2008



Be able to choose to deliver by *value or date*  
*"Trim the Tail"*



Slide 23

©Alistair Cockburn 2008



In the 21st century, software engineering will use  
*craft, cooperative game & lean principles*

***Craft***

developing skills in a medium  
*shu - ha - ri* progression

***Cooperative game of invention and communication***

teamwork, communication, strategies

***Lean processes*** (“unvalidated decisions = inventory”)

small queues, cross-trained people, ...

***Design as knowledge acquisition***

early integration  
pay to learn early  
trim the tail at the end

Slide 24

©Alistair Cockburn 2008



<http://Alistair.Cockburn.us>

