



## Bits And Mortar

Ric Parkin

ACCU Conference 2008



Sometimes lessons, rules and processes learnt in one field can be applied to a different field

Is there a field that Software Development has learnt from?

Architecture

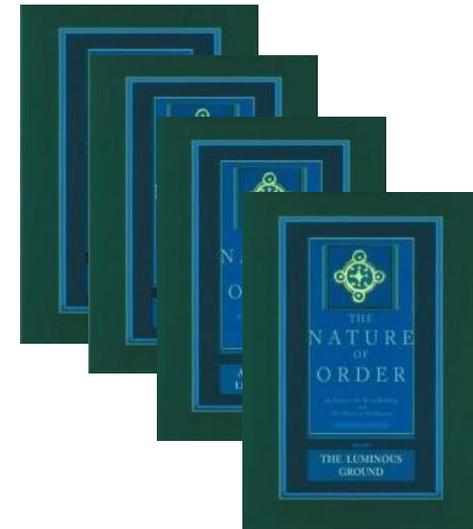
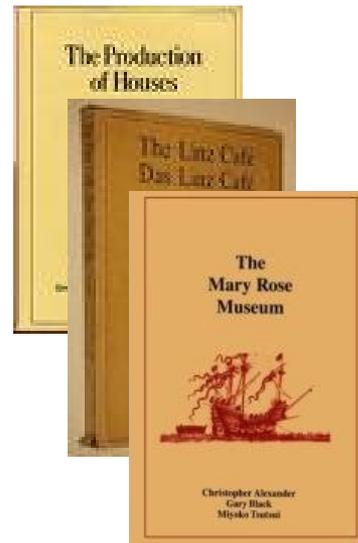
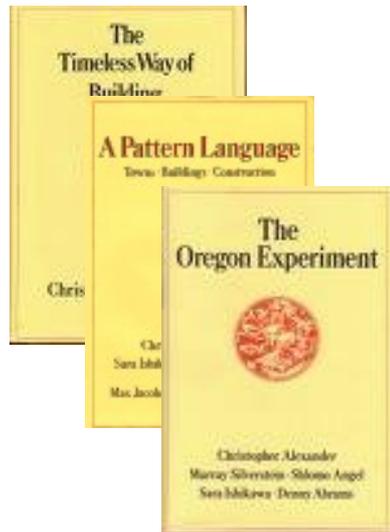
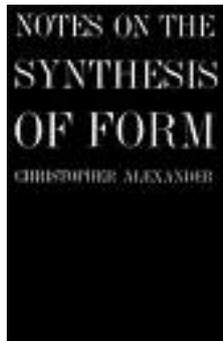
Can we learn more from that field? Or others?

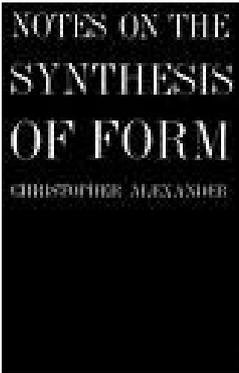
And can we apply *our* knowledge to that field?



Mathematician and Architect

<http://www.patternlanguage.com/>





From Phd  
Thesis

Idea is that design is about achieving a number of requirements

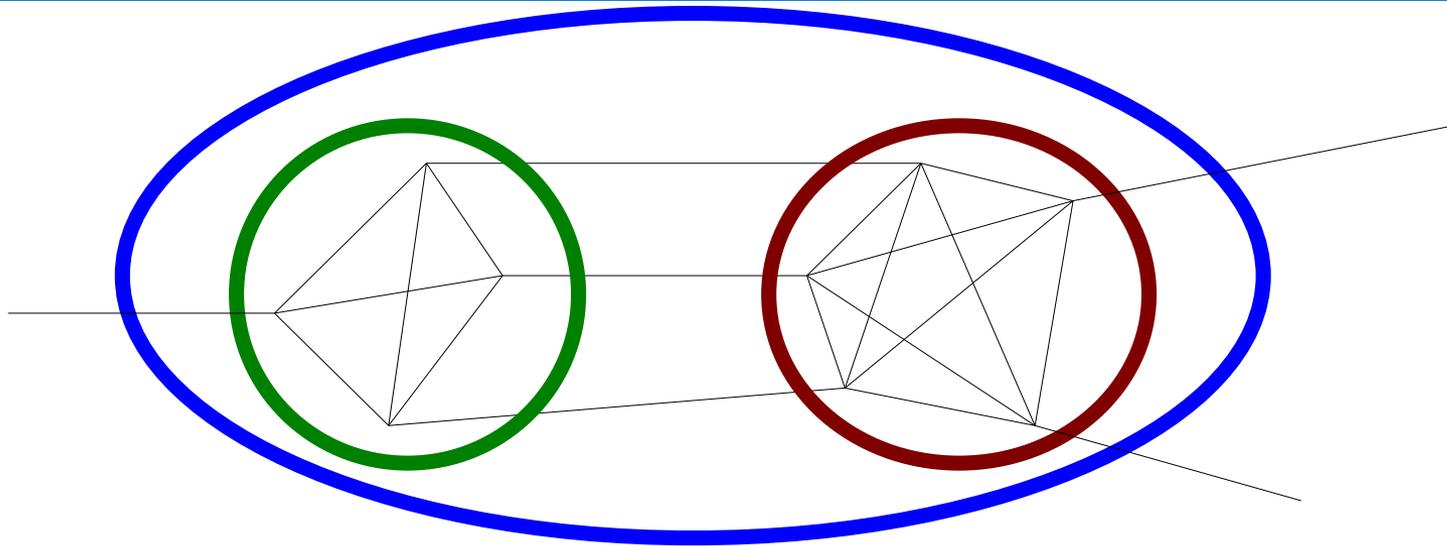
- Eg Cost/Strength/Quality/Usability
- Kettle Example: 21 different goals

Too hard to solve all at once – attempt to solve one thing and others will fail

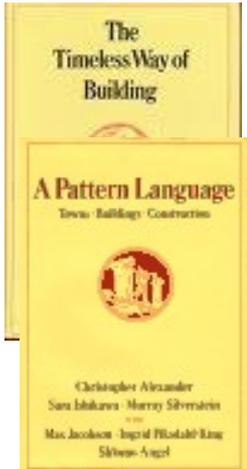
- Eg Make vacuum walls – safer and more efficient, but costs more and reduces market

How to simplify problem to manageable scale?

# Islands Of Connectedness



- Changing one influences others
- Connections and weights
- Reduce search space by finding *relatively isolated* islands of interconnected forces, and recurse
- Solve independently - “Diagrams”
- Piece together – blocks of design



The Timeless Way Of Building

A Pattern Language: Towns, Buildings, Construction

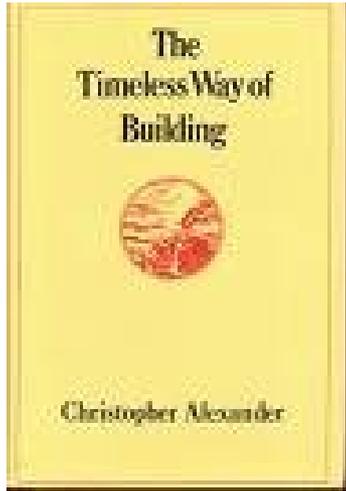


The Oregon Experiment

The Production of Houses

The Linz Cafe

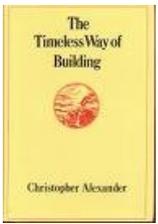
The Mary Rose Museum



“Notes” has method of discovering diagrams to solve for arbitrary systems

But in practice you can find solutions *in advance* – because networks of requirements *recur* in real life

Solutions are called *Patterns*



The character given by interlocking patterns to form a harmonious “whole”

Some aspects:

Eternal

Comfortable

Alive

Stable

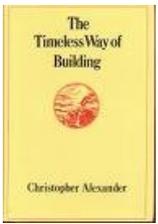
Sustaining

Whole

Egoless

Exact

Free



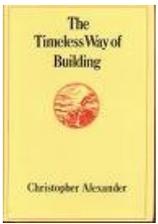
Design tries to accommodate various forces

Resolves internal tensions

Requires a network of interconnecting patterns at differing scales to balance each others' consequential forces

*When* something changes, the system is likely to have other parts that can help in balancing the new forces caused by the change.

Like a rich ecosystem



A place is given a character by the events that keep on happening in and around it

- Lecture theatre
- Punting
- Bustling market square



“Watching the world go by” is intimately linked to the place where you do it, eg a porch

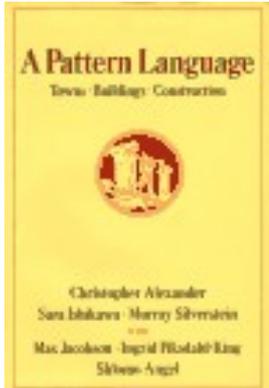
You have a porch, so you use it and watch the world

You want to sit out and watch, so you build a porch

Exact solution has *cultural* input too

Software analogies? User's tasks and habits; input data; hardware events; flow of data and events through a system

# A Pattern Language: Towns, Buildings, Construction

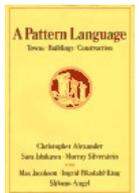


Lots of patterns at different scales

- Regions, Towns, Communities
- Areas, Buildings and Rooms
- Construction
- Decoration and furniture

Remarkably readable, or can dip in and out

Full of classic pattern “A-ha” moments when you recognise a pattern you know, or realise why something that annoys you doesn't work



Is a very literary type of pattern, but does have various sections

- Name and Star Rating
- Evocative Picture
- Introduction, including which forces this helps complete
- Problem Context (headline and discussion)
- Solution (headline and discussion)
- Illustrative Sketch
- Consequential forces arising, including which patterns may be of help next

Language because patterns lead from one to another

- incoming and outgoing consequences: sentences get formed

Examples...houses, offices, organisations



- Fireplace
- Window with a view of life
- Waist-high shelf
- "Corridor" through rooms
- Sunny sheltered spot
- Six foot balcony – get a table in
- Balconies half-in-half-out – public/private rooms
- Pools of light
- Climbing plants



## Alexander – Office Patterns (unpublished)

- Reception Welcomes You
- Small Work Groups
- Water cooler places. Informal discussion areas. Whiteboards!
- Personalised desks
- Amount of communication vs distance apart
- Different rooms, floors, buildings
- Shielding from distractions, vs overheard conversation. Half Open Office
- power/network connections are changable

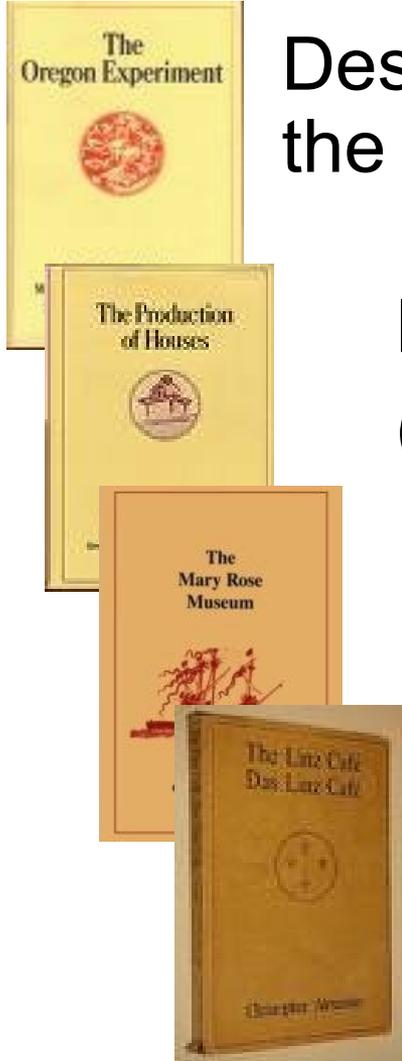


Some ideas...

- Conway's Law
- Dedicated Project Teams
  - May be temporary
- Teams sit together
- Make Progress Visible
- Regular Discussions
  - Continuum of Regular/Small/Quick -> Occasional/Large/Long



# How Does It Work In Practice



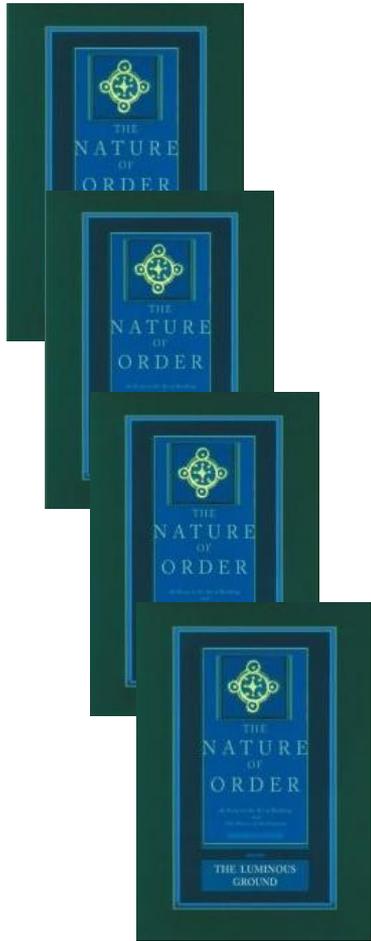
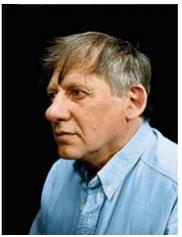
Designing and building the campus of the University of Oregon

Building a group of houses in Mexico (+ describes a theory of house building)

Designing the museum to house the “Mary Rose” - “agile” design

2 **The Linz Cafe (Center for Environmental Structure Series)**  
 by Alexander, Christopher  
**Binding:** Hardcover **Publisher:** Oxford University Press, USA **ISBN-13:** 9780195202632 **ISBN:** 0195202635  
**Description:** Very Good. 0195202635 Clean, unmarked pages. Cover and jacket in very good condition. Jacket protected by mylar. [read more](#)

**add to cart**  
**price: £578.26**  
 Ships within 2 weeks  
**add to wishlist**



The Phenomenon of Life

The Process of Creating Life

A Vision of a Living World

The Luminous Ground



Parts of Christopher Alexander's work has already had an influence on software

- Patterns and Pattern Languages

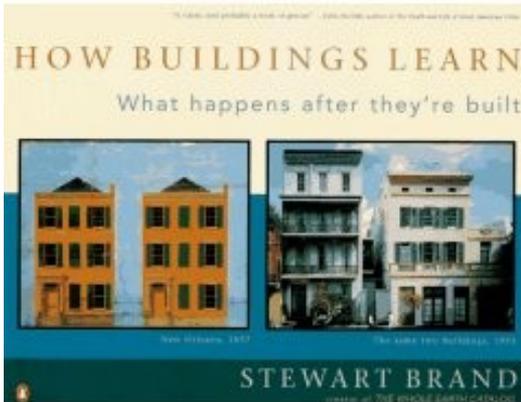
Some other parts could also be ripe

- Agile design ideas
- Pattern discovery methods from “Notes”?
- The Nature Of Order (will let you know)

Is there a deep analogy at work here?

Notes only mentions “design”

Other ideas from the world of architecture?...Time



## Stewart Brand *How Buildings Learn: What happens after they're built*

*Whole Earth Catalog (1968), Hackers Conference (1984)  
(Coined the phrase "Information wants to be free"), The  
WELL (1984), Global Business Network (1988), and The  
Long Now Foundation (1996)*

*"Many seem to treat it as a book about systems and software design."*

On virtually every page there's something that reminded me of software development.

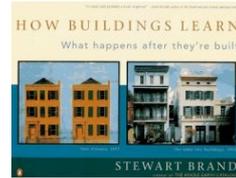
Keep this in mind as I introduce some of these ideas

Places with more info

<http://sb.longnow.org/Home.html>

[http://en.wikipedia.org/wiki/How\\_Buildings\\_Learn](http://en.wikipedia.org/wiki/How_Buildings_Learn)

[http://www.gyford.com/phil/writing/2004/10/24/how\\_buildings\\_le.php](http://www.gyford.com/phil/writing/2004/10/24/how_buildings_le.php)



*“Flow, continual flow, continual change, continual transformation”* Rina Swentzel on her home village

Buildings almost never adapt well, but they *always* adapt because of the changes around them

*“Form ever follows function”* - Louis Sullivan 1896

- Modernist founding idea.
- **Relies on fallacy that we can *anticipate* function**

*“We shape our buildings, and afterwards our buildings shape us”* - Churchill

- Eg Parliament – deliberately small, two sided room
- But missed “...then we shape them again, ad infinitum”



## Change forces change

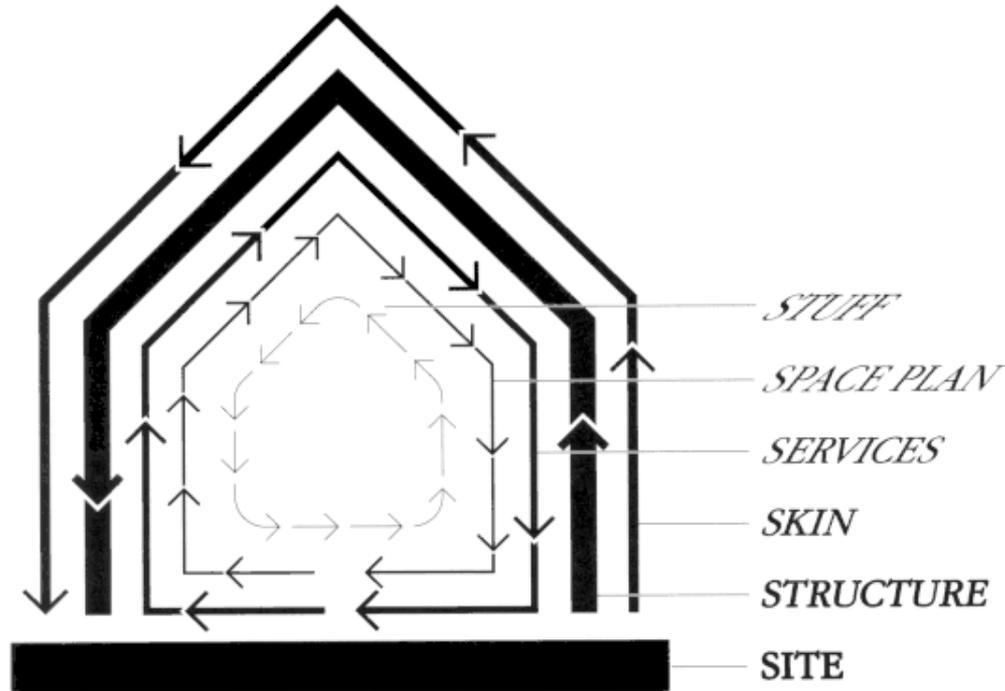
- social changes forced kitchens to change from servants lair to middle class entertainment focus;
- garages built for cars (and later for computer start-ups)
- introduction of the TV
- change from long-term nuclear family – still adjusting

More is spent maintaining old buildings over time than building new ones; eg Norwich's churches

## Three forces influence choice of solution

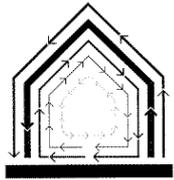
- Technology, Money, Fashion

*“We are convinced by things that show internal complexity, that show the traces of an interesting evolution” - Brian Eno*



Duffy - "Our basic argument is that there isn't any such thing as a building. A building properly conceived is several layers of longevity of built components"

*Separation of Concerns In The Time Axis*

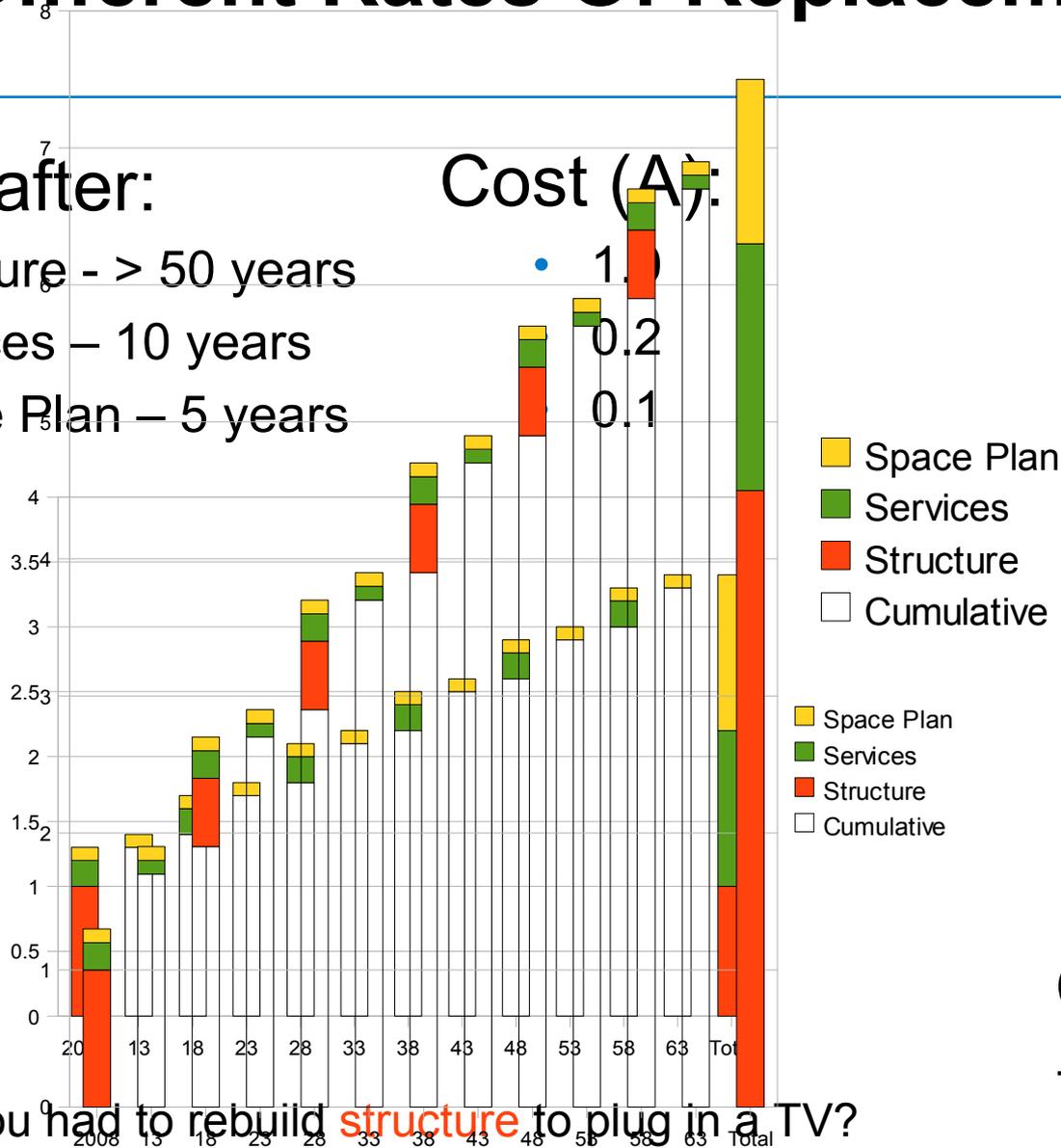


Replace after:

- Structure - > 50 years
- Services – 10 years
- Space Plan – 5 years

Cost (A):

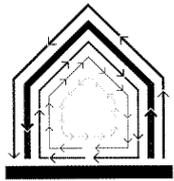
- 1.0
- 0.2
- 0.1



Cost (B):

+ half the next layer

What if you had to rebuild structure to plug in a TV?



[http://systemicbusiness.org/pubs/2000\\_IBM\\_RC21694\\_Simmonds\\_Ing\\_Shearing\\_Layers\\_Info\\_Sys\\_Dev.html](http://systemicbusiness.org/pubs/2000_IBM_RC21694_Simmonds_Ing_Shearing_Layers_Info_Sys_Dev.html)

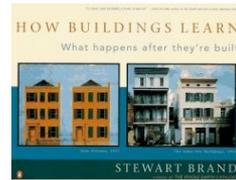
*“In this paper we respond to the observation that systems are subjected to qualitatively different scales and rates of change, and should consequently be constructed to adapt in “shearing layers.” This observation applies equally to social systems such as business (or other) enterprises, and to the software systems that they use.”*

Software is written to meet a set of requirements – from the users and their needs, and the wider environment.

**All requirements change *but at different rates***

Fundamental requirements change very occasionally; more simple things change more often; small details rapidly

Conway's Law – team communication boundary adapts *slower*



- Quick
- Cheap
- Unplanned
- Adaptive
- Small
- Incremental
- Short term horizon
- Disposable

Artists in lofts; Factory warehouses; Small business units – expand into next door (and back); Technology incubators; MIT building 20; Small houses; Extensions

We have a track record: computer start-ups in garages and sheds

- Apple, Hewlett-Packard, DisplayLink

# Low Road Examples



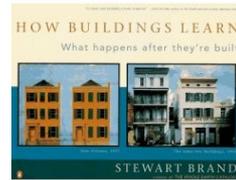
Daniel Friedman saw a splendid adapted mobile home in Rhinecliff NY, north of Poughkeepsie where IBM has a factory and research lab. Picture at <http://www.inspect-ny.com/structure/mobileinspections.htm>

I found a joke site “Mobile Homes Of Mississippi”, but has some fascinatingly varied homes and adaptations: <http://www.drbuk.com/gmhom/gmindex.html>



Quick, Cheap, Unplanned, Adaptive, Small, Incremental,  
Short term horizon, Disposable

- Shell scripts, batch files
- Command pipes
- Simple command line tools to do a job
  - Eg Build scripts, Update Version Number script
  - Details are often unique to *you*
- Office Automation
- Macros
- Emacs?
- Hardware: Cambridge Coffee Pot



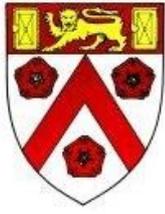
- Slow change
- Long term commitment
- High quality to “set the standard”
- Planned
- Constrained (but adaptive within)
- Big vision
- Main building phase, then incremental change, occasional rework

Chatsworth House

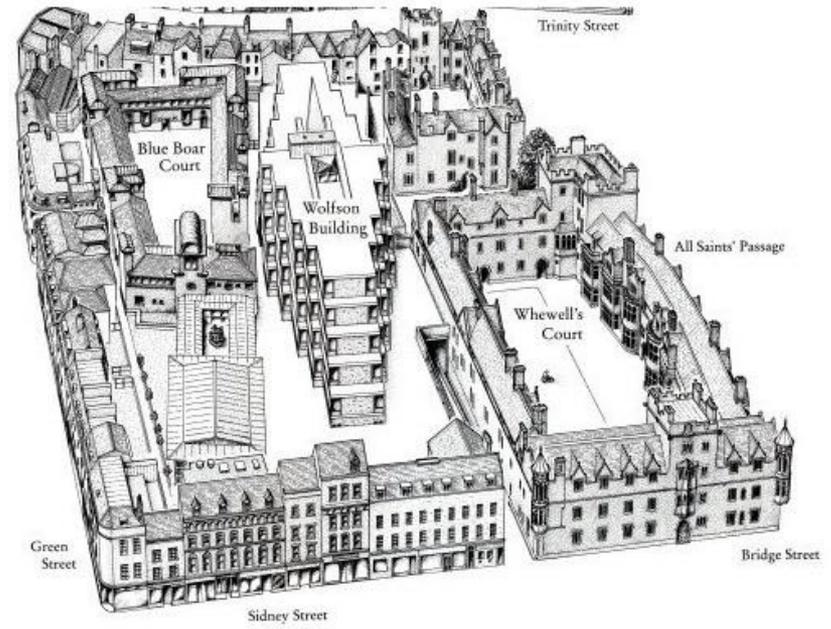
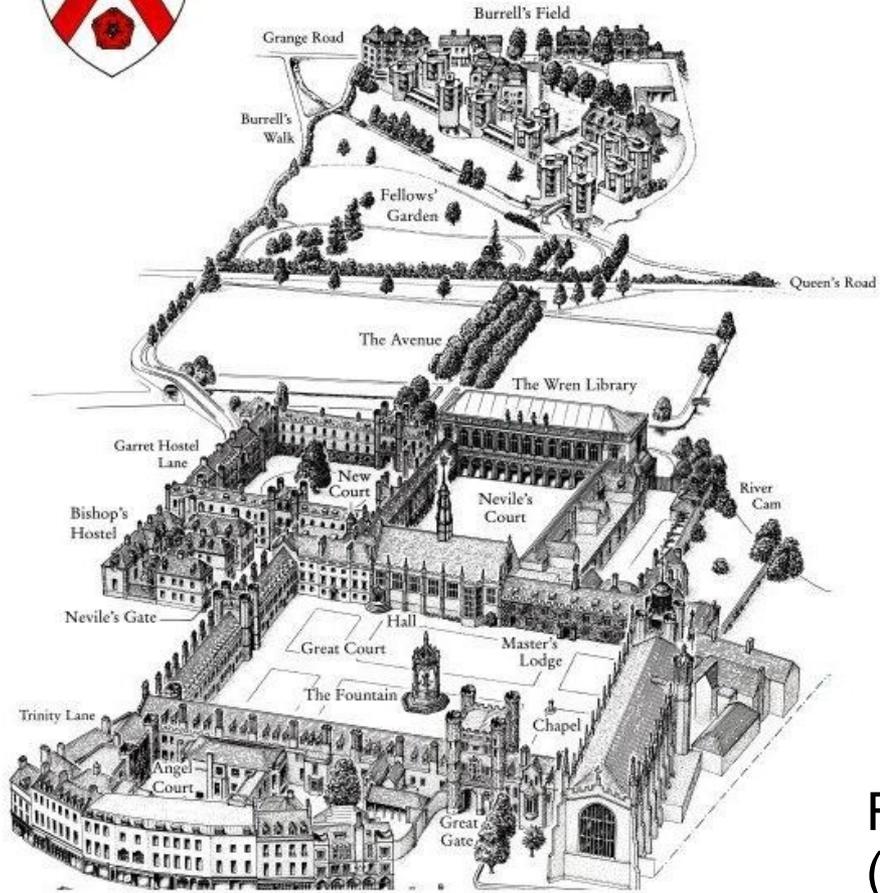
Presidential Houses: Mount Vernon, Montpelier, Monticello

London Library, and Boston's Athenaeum

# High Road Example



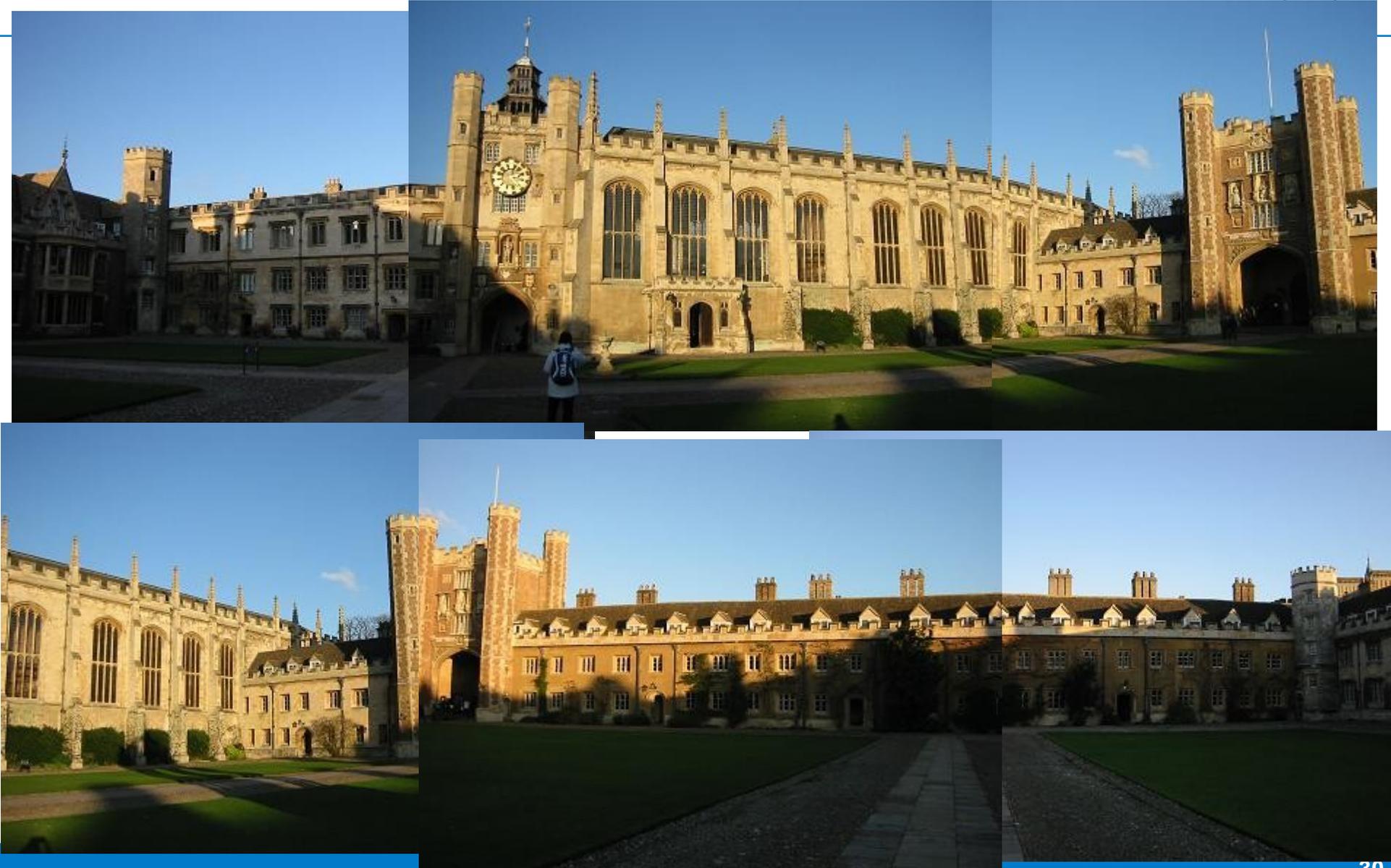
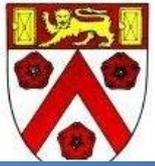
TRINITY COLLEGE  
CAMBRIDGE



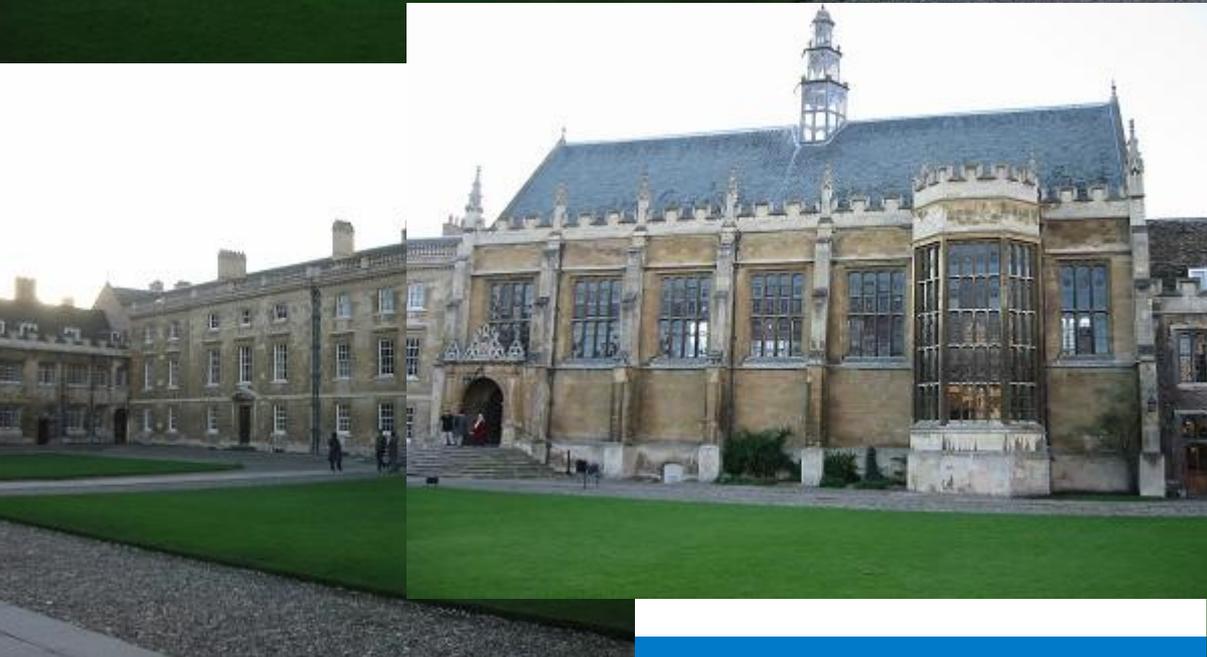
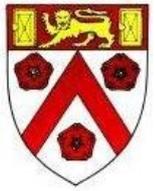
©Trinity College, 1996  
Drawn by Jeremy Bays, Art-Work-Shop, PO Box 389 Cambridge CB3 0SA.

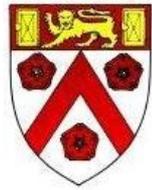
Founded in 1546 from Michaelhouse (1323) and King's Hall (1317)

# Great Court N and E Ranges



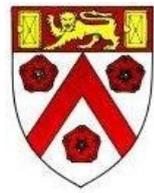
# Great Court S and W Ranges



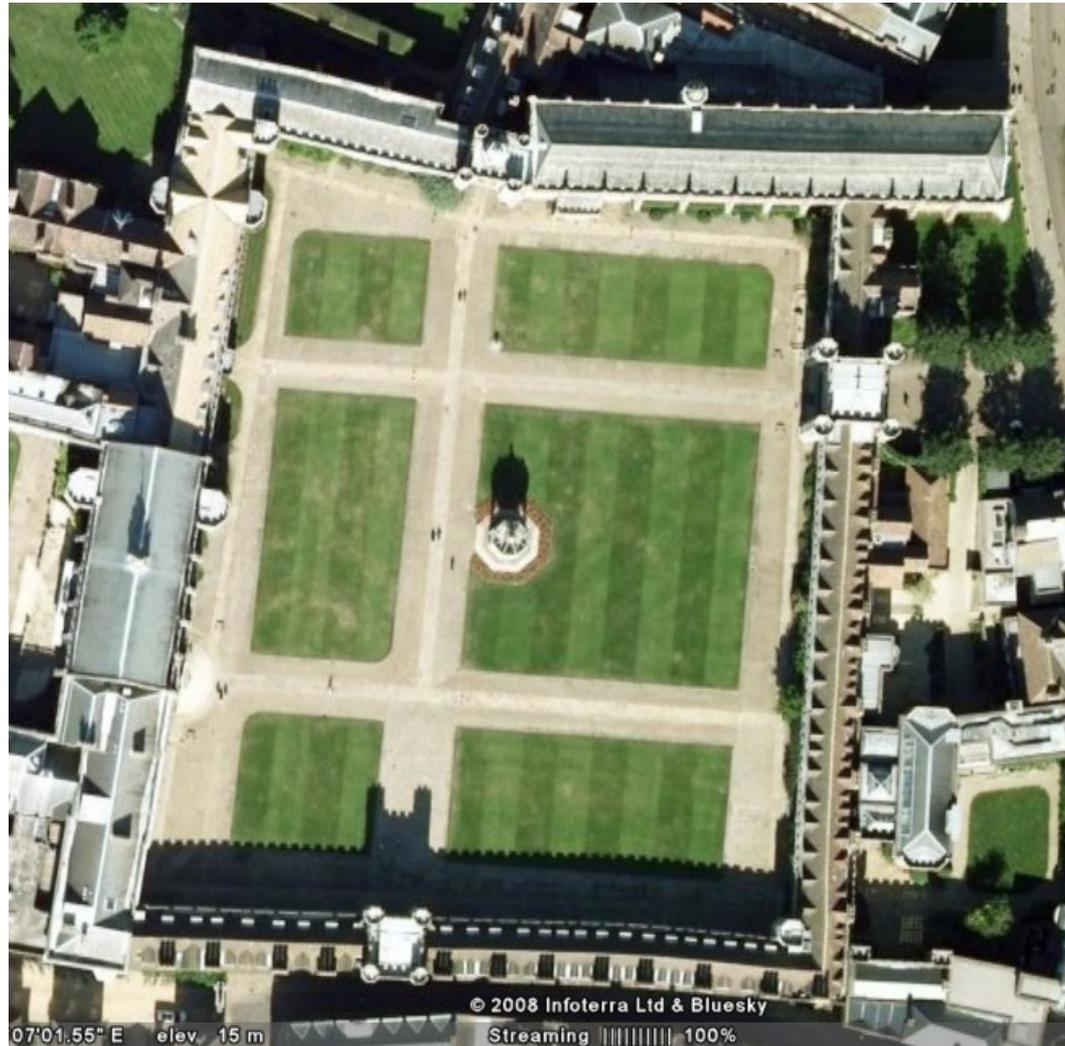
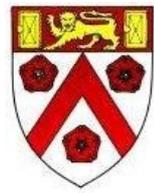


- King's Gate dated from 1428-32
- Nevile's plan for Great Court around 1600
- Moved tower back 30m to fit between the library and the new chapel. New floor and clock added in 1610. Is a real squeeze, and range is not square.
- Restoration in 1988 reported evidence of it being rather carelessly and hastily reconstructed, and is something of a “lean-to” against the chapel, and blocks the west window

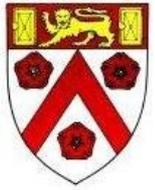
# High Road Tweaking



# More High Road Tweaking



# High Road Long View



Avenue of Lime Trees

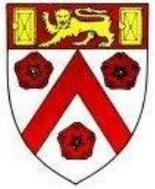
Replacements



- 1994 Doom patch - The Unholy Trinity  
Steve McCrea, Simon Wall, & Elias Papavassilopoulos



[http://www.youtube.com/watch?v=j1u6QZ\\_KiXU](http://www.youtube.com/watch?v=j1u6QZ_KiXU)



Slow change, Long term commitment, High quality, Planned, Constrained (but adaptive within), Big vision, Main building phase, then incremental change, occasional rework

## Most commercial software

- Long term need; Upgrades for support agreements
- Architectural vision; long term developers (!!!)
- Main development, incremental improvements, occasional major reworking

How does Agile development fit in?

Pushes us towards Low Road adaptivity

- Small scale tends towards Low Road
- Larger scale High Road structure, Low Road tweaks



Note that Low and High Road are defined in terms of how they change over time to adapt to changing needs

No Road architecture steps outside such petty concerns and *just doesn't work or adapt very well*

Lots of bad Modern Buildings

- Leaky roofs
- Inflexible rooms, layouts, etc
- Services built-in – shearing layers fused
- Technological “fixes” that don't work

# T5



Problem of architecture as “Art” - doesn't feel it has to be useful or usable

Art must experiment; most experiments fail

Art is of the moment; the moment passes; but might come back as retro!

Awards are based on photographs, not use

- Pretty façade not practical interiors

*“If a pleasure-giving function predominates, it is called art; if a practical function predominates, it is called craft” - Henry Glassie*

Art vs Craft vs Engineering

*Discuss*



Speculator commissions; Architect designs, adjusted though planning permission; specialist engineers plan; building contractor + subcontractors build; occupied by Facilities Manager, Landlord and Users

Something gone wrong, want to change something?  
Who do you talk to?\*



*“You never go back. It's too discouraging”* - anon architect

Pay architects a retainer to come back

- Make it worth them to say “No, not yet”
- Make it worth it to them to make things changeable
  - Avoid the obsession to get it right first time

## Post-Occupancy Evaluation

Learn from your successes and failures

One-stop design-build-manage firms

Eg <http://www.kajima.co.jp>

Collaboration and long-term commitment by the same *team*. When problems occur, single point of contact and can talk to the architect/engineer etc and adjust

Life-cycle management: *“Today the most salient trend of commercial properties is the shift from the so-called “scrap and build” to operations and maintenance.”*



- Large IT projects
  - Very hard to change the design if it doesn't work in practice
- 4G Languages?
- Specialisations without teams – “over the wall”
- “Magazine Technology”
  - Looks good in articles, demos, and marketing presentations, but does it actually deliver
- Waiting for ~~Godot~~ Longhorn
  - “Nothing is delivered. Twice”



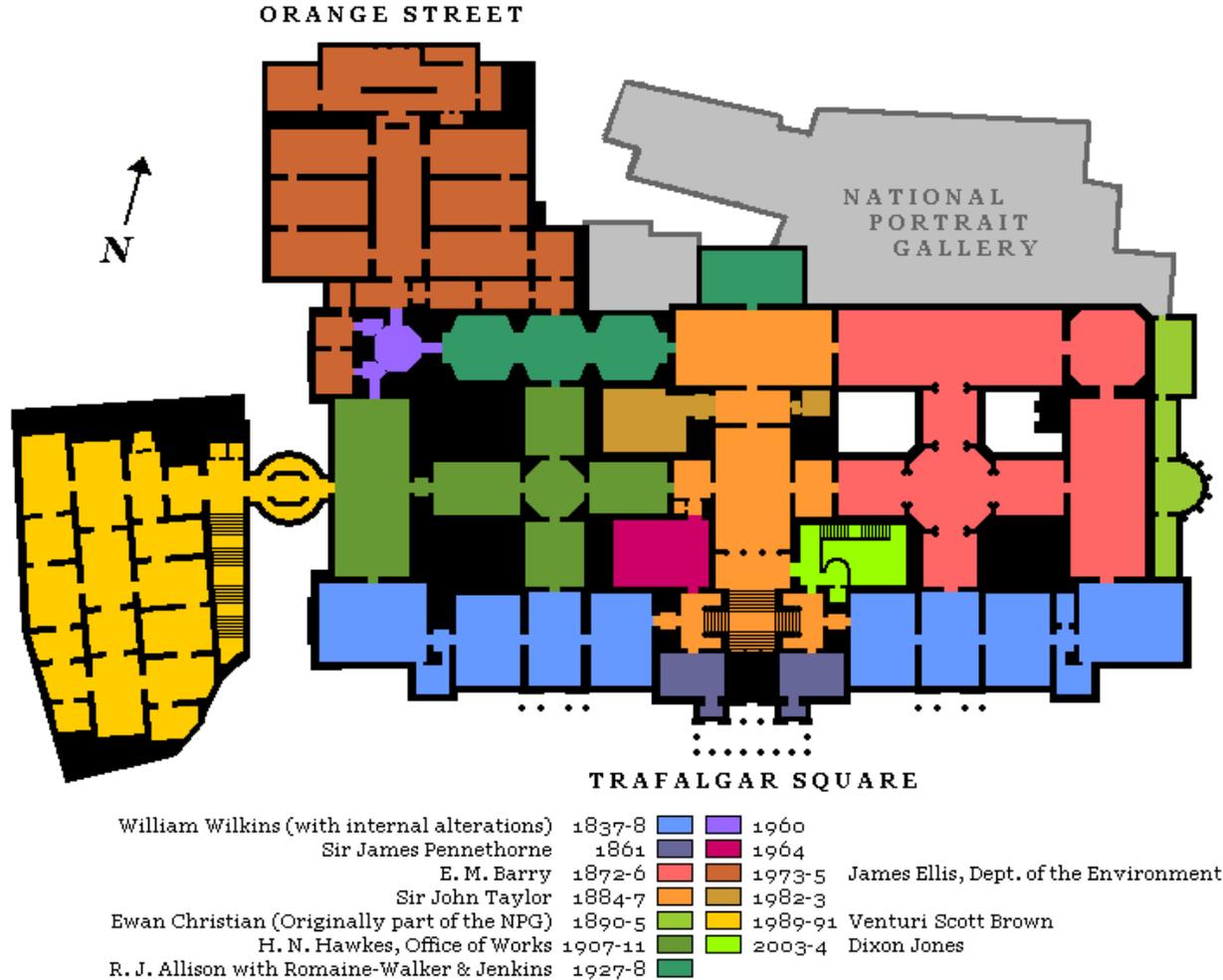
## Alexander – The Oregon Experiment

Instead of a large-lump development (“replacement”), do large and small scale changes over time (“repair”)

Allows learning from what **works** and what **doesn't** – accepts mistakes will happen, and allows them to be repaired

“Maintenance *is* learning”

# Piecemeal Growth Example



[http://commons.wikimedia.org/wiki/Image:National\\_Gallery\\_1st\\_floor\\_plan.gif](http://commons.wikimedia.org/wiki/Image:National_Gallery_1st_floor_plan.gif)



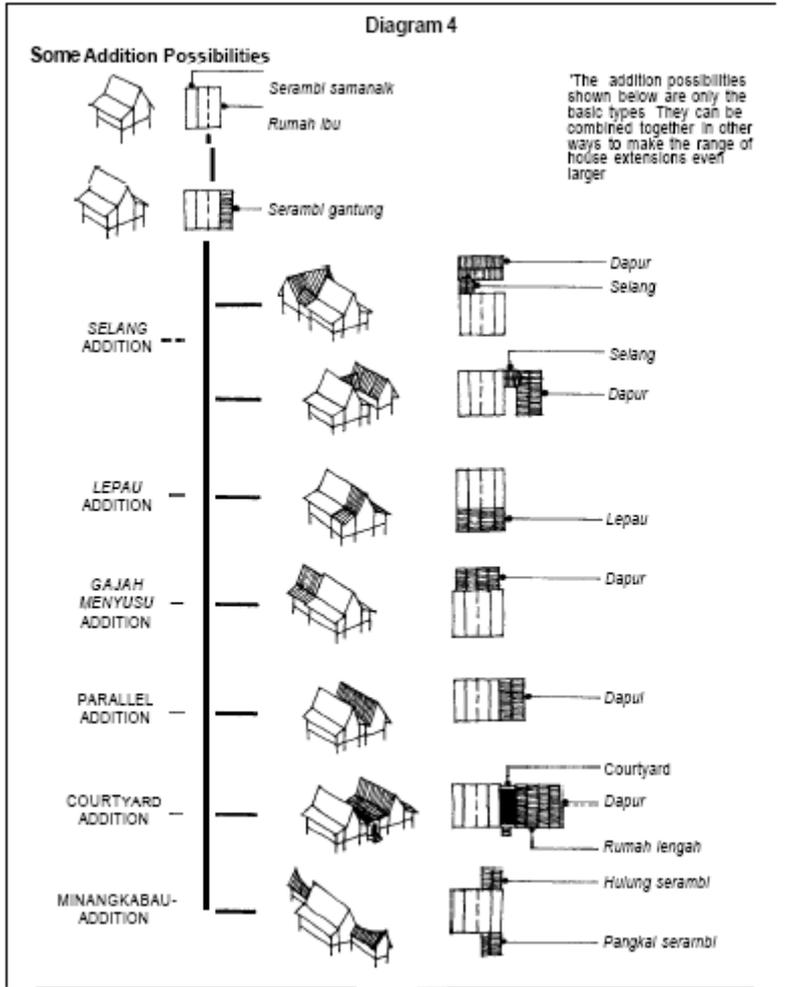
Doing something new gives plenty of opportunities to get things wrong

Take a shortcut through the design process by copying from what's around you that works

- Pre-constrains the solution – makes it more manageable
- Fits in and is familiar
- Patterns are pre-canned solutions



- User Interfaces
  - Quite a bit of fashion
  - “I want one like that”
- Do things in the One True Style
  - Python
- Do things in your local style
  - STL or own library
  - My Handle generator template
  - Asynchronous thread engines
    - Requests in via message queue
    - Raising events
- Patterns are pre-canned solutions

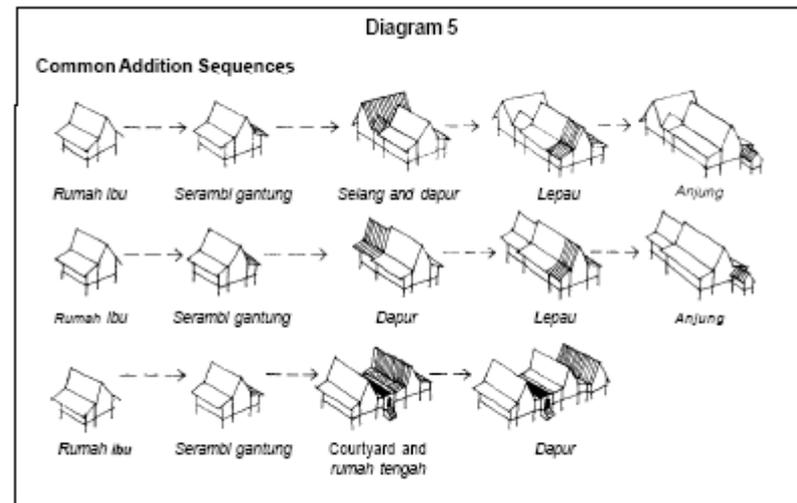


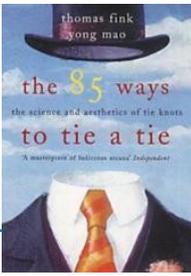
*▲ Addition possibilities*

Lim Jee Yuan

## *The Malay House*

Patterns plus an order to apply them generates a house





## Generative Pattern Languages

- Tiny pattern-steps
  - Different orders create different designs
- Generate all possible using random walks
  - 85 ways to tie a Tie
  - Richard Harris' knot article in Overload 84

Incremental releases

New features added on

Get value in stages

Frameworks?

Dynamic Addons and plugins



Satisficing is a decision-making strategy which attempts to meet criteria for **adequacy**, rather than to identify an **optimal** solution

Cybernetics - optimise costs, including the cost of working out the solution

Doesn't try to solve problems **perfectly**, it fixes them **just enough** ...and keep on **adjusting** and making a bit better each time

Changes *encode* the needs and aspirations of the occupants *over time*

Evolution, adaptation.



- Multiple clients can pull your software in different directions – how are you going to reconcile them?
- Modest tweaks – better (for now) but check risk
- Steps that improve bit by bit
  - But you need to accept when it needs radical overhaul
- Accepting Technical Debt
  - Pricing models; Discounting; Opportunity cost
- It's already happened:
  - Software archaeology
  - Debugging - “Why did they do that?”

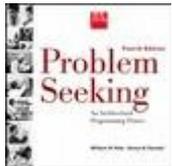


All design is prediction

All predictions are **wrong**



Architects do “Programming” - talking to users to find out what problems they'll need to solve



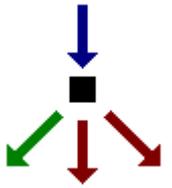
Peña, Parshall, Kelly - *Problem Seeking*

Good idea! **But...**

Many buildings are brilliant solutions *to the wrong problem*

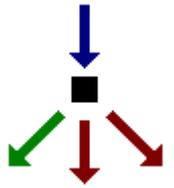
Good intentions, but do the users actually know what will happen? “Whatever the client says will happen, won't”.

Or wishful thinking - “put in fibre-optics for when broadband comes in”, and wireless makes it obsolete

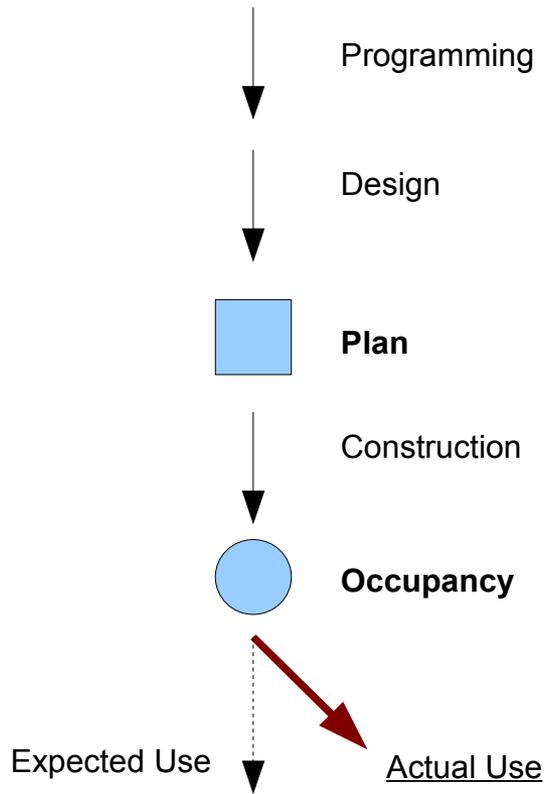


## Solution?

- Don't make a plan, make a *strategy*
- “Accommodate perversity”
- don't design for an expected future, design for a *range* of futures
- Scenarios explore the problem domain
- Wild scenarios question assumptions

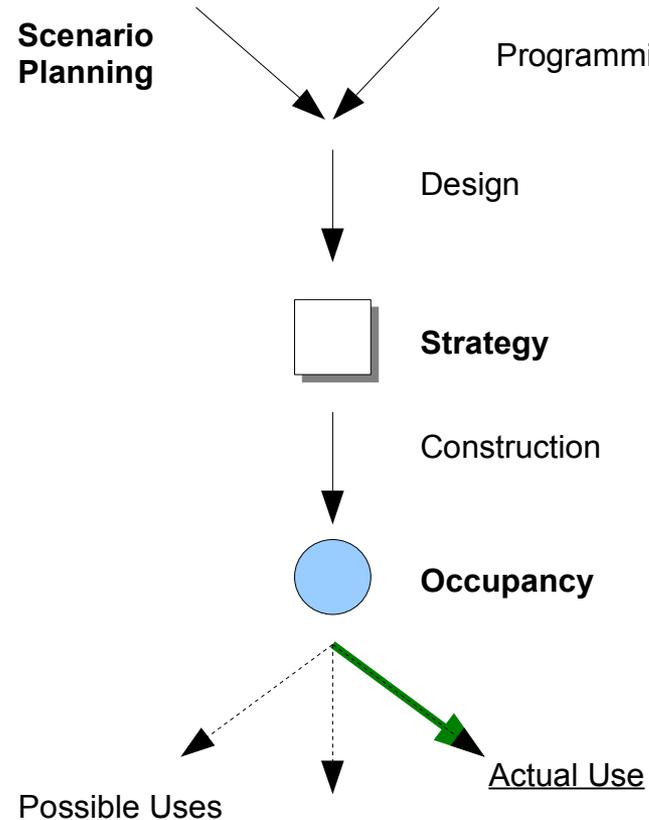


## Traditional Building

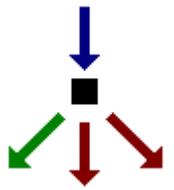


**“Wrong!”**

## Scenario-Buffered Building



**“Oh, that one”**



- Interview people to pick up vocabulary, major issues, and expectations of the future
- Find focal issue - "what keeps you awake at night"
- Explore "Driving Forces", and rank them in importance and uncertainties
- Add reliable "predetermined elements"
- Identify "Logics" - basic plot lines
- "Think the unthinkable" introduces novelty
- Devise a strategy to accommodate *all* scenarios
- "Robust" vs "adaptive" strategy
- Find "Early Warning" indicators of failures



- Chess – favour moves that increase options; shy from moves that end well but require cutting off choices; work from strong positions that have many adjoining strong positions
- Buildings – overbuild structure for future expansion; excess service capacity; separate high and low volatility areas; use local materials – easier to match or replace; medium size rooms are most flexible; add storage

*“We overestimate technology in the short term and underestimate it in the long run”*



- Loose fit buildings – expand into next door
- Not enough time, or money? Some highly planned and finished areas, some “raw” - unfinished but usable, to be finished later according to need and money
- A building is not something you finish. A building is something you start.
- “Time is the greatest innovator” - Bacon
- “Pave where the paths end up”



- Start conservative, become radical
  - You can specialise from the general, not vice versa
- Continual house
  - Build a small but liveable core for the cost of the normal down-payment
  - Instead of paying interest, invest that money to grow the house bit by bit
  - Grows with you in the long run
- Flexible space
  - Victorian terraces
    - wide hallway allows many different adjustments
    - generic rooms opening onto each other & hallway
    - avoid overspecialisation



No one can visualise how it will look and feel

Construction ought to be a long process of cut-and-try.

Alexander *“You are constantly finding out about the building while constructing it, and what you will find out is inherently and necessarily unpredictable”*

Failing small, early, and often, leads to succeeding long term and large scale

Maintenance, correction of faults, and improvements should all blend together

Do something for more than one reason

- Immediate purpose
- Serve larger goal of “Healing the whole”
- Prepare for next improvement



Japanese artistic virtue - *“The recognition that in a beautiful thing there is always some part which is lovingly and carefully done, and some parts which are very roughly done”*

Leave some things unfinished

Ready for unexpected uses

Once things are “finished” still lots of tinkering to get things right

Distrust “optimal” solutions

Need a margin of error for surprise and change



Synchronic

How things fit together *now*

Diachronic

How things developed *over time*

People tend to think Synchronic

Diachronic is almost always ***more revealing***

We have lots of tools for the former, very few for the latter

- Check-in comments and change logs
- Subversion “Blame” - Perforce Time Lapse View
- “Animated” UML?
- Apple's Time Machine

# One Offs And Change

Software development is about designing something *new* each time

- Not always - some fields are well understood. But they lend themselves to automatic generation...using a new tool that needs to be written

Requirements often change, or your understanding of them does

Bug-fixing, introducing new features, and changing old ones are a major ongoing process in software

Rarely throw away and restart – too expensive

“Don't Scrap, Adapt”

Just Like Buildings



Software design and building design seem to be very similar. Why?

- Design something new, not mass manufacture
- Too complex to get right first time
  - Synthesis Of Form – applies to *any* design
- Adjust rather than start anew
- Requirements change over time - Shearing layers
- Low Road/High Road styles and approaches
- Is never finished – adapted and extended as needs change

Similar *design* and *temporal* dynamics in Architecture and Software Development result in a deep and productive analogy

This is why there are so many ideas that can be transferred between them

What software design technique will *you* apply to to your next building project?



## Alexander – The Nature Of Order, Vol 1 “The Phenomenon Of Life”

*If you accept the view of order that I am presenting, you will find it has unexpected intellectual results. It modifies our view of the physical universe and the way the universe is put together. Thus, what starts out as a way of understanding architecture ends up, also, as a view which may affect our understanding of physics and biology.*

PS. Dan Cruickshank's Adventures in Architecture 9pm BBC2  
Grand Designs 9pm Channel 4