# five considerations

kevlin henney | 23rd april 2005 | accu conference | oxford

## context

| | |
|---|---|
| five considerations | (in answer to a non-trivial question) |
| four green fields | (recommended by alan o'callaghan) |
| three conference delegates | (charles weir, frank buschmann, me) |
| two days into a conference | (oopsla 2001, tampa, florida) |
| one pint of guinness each | (to start with...) |

no partridges or pear trees in sight

see http://www.artima.com/weblogs/viewpost.jsp?thread=5432

# economy

## a quote

"There have always been fairly severe size constraints on the Unix operating system and its software. Given the partially antagonistic desires for reasonable efficiency and expressive power, the size constraint has encouraged not only economy but a certain elegance of design."

Dennis Ritchie and Ken Thompson

## less code, more software

software functionality is not simply correlated to the amount of code

- a desire for more code is all too often, alas, fulfilled

economy of expression often yields other economies

- performance, footprint, comprehension overhead, cost of change

so remove to improve

- modification is not necessarily additive: iterative and decremental development
- cutting back encourages growth and longevity

but the goal is not to turn your code into line noise

- don't encode your code

# symmetry

## a definition

***symmetry*** the quality of being made up of exactly similar parts facing each other or around an axis.

- correct or pleasing proportion of the parts of a thing.
- similarity of exact correspondence between different things.

The New Oxford Dictionary of English

## the course of symmetry

symmetry is a form of balance and regularity

- balancing parts to form a memorable whole, easy to recall and use

introducing and preserving symmetry can simplify structure and use

- JUnit's assertion methods, lexical_cast's handling of spaces and numerics, hierarchical object ownership versus asymmetric or shared, edit undo and redo

symmetry breaking can be profound and provocative... or just broken

- "When in doubt, make it symmetrical" (Christopher Alexander)

beware of overdoing it: there are many superficial symmetries

- setters with getters, V-model testing, interdependent base and derived classes

# spacing

## a metaphor

"At the most elementary level, a building is a construction of physical elements or materials into a more or less stable form, as a result of which space is created which is distinct from the ambient space."

William Hillier

## separating concerns

spacing applies as much to components as it does to the layout of code

- e.g. use of interfaces to partition code "neighbourhoods"

express independent ideas independently

- e.g. the orthogonality of generic programming

insufficient spacing leads to code that is too dense and intertwined

- e.g. the confused cohesion of realloc

too much spacing makes things unnecessarily sparse and fragmented

- code can drown in whitespace
- a component structure can look more like an archipelago than an assembly

# visibility

## an observation

"The cause is hidden. The effect is visible to all."

Ovid

## discrete and discreet

software is digital, discontinuous and aphysical

- software development is informational engineering rather than physical

this wrong-foots much of our intuition drawn from the physical world

- size and time estimation, perception of structure, comprehension of failure

use of physical space for exposition and representation is metaphorical

- e.g. structural diagrams that show objects, classes, packages, components, etc

practices need to emphasise making the invisible more visible

- iterative and incremental micro- and macro-processes, scenario-driven development, test-driven development, patterns, use of idiom, consistency

# emergence

## an example

Flocking is a common demonstration of emergent behaviour.... The result is alike to a flock of birds, a school of fish, or swarm of insects.

Basic flocking is controlled by three simple rules:

1. *Separation* avoid crowding neighbours
2. *Alignment* steer towards average heading of neighbours
3. *Cohesion* steer towards average position of neighbours

With these three simple rules, the flock moves in an extremely realistic way, creating complex motion and interaction that would be extremely hard to create otherwise.

Wikipedia

## the product of the parts

simple rules and mechanisms can lead to sophisticated behaviour

- not all aspects and levels of a system's behaviour have to be defined painstakingly and explicitly, enumerated in patchworks of special cases

behaviour can arise out of combination and collaboration

- it does not itself need to be modular in definition

there is an element of giving up control to attain it

- e.g. polymorphism versus explicitly spelt out if-else or switch statements

however, emergence is not magic and should not appear muddled

- emergence may be in tension with consideration of visibility

## conclusion

it was a good drink and a great conversation

- the original question may or may not have been answered, but it was definitely worth having, as was the beer and the discussion

the considerations provide useful headings and starting points

- they are not an end in themselves
- considerations are neither recommendations nor rules