Software Engineering Methodologies

Dr. Julie Greensmith
julie.greensmith@nottingham.ac.uk

Coming up today....

- Fundamental definitions
- Emergence in software engineering
  - coping with the development of complex systems
- Software Engineering Methodologies
  - common practices in software engineering

Jargon buster!

- Programming vs compsci vs software engineering
- Software, Software process, software process model
- Methodology, agile, emergence, iteration, prototype

What is "software"

- Easy - not hardware! :-) not quite that trivial
- *Computer programs **and** associated documentation such as requirements, design models and user manuals developed for a particular market*
- New software can be created by developing new programs, configuring generic software systems or reusing existing software

Thus, software engineering is...

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- *"Software engineering is concerned with principles and methods for specifying, designing, implementing and maintaining large software systems." R. Sirewalt 2004*
- *Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.*

Spot the difference

- Computer science is concerned with theory and fundamentals
  - concerned with the practicalities of developing and delivering useful software.
- Computer science theories are still insufficient to act as a complete underpinning for software engineering because of *users, organisations and complexity*
- This is unlike e.g. physics and electronics


COMPUTER SCIENCE IS NO MORE ABOUT COMPUTERS
THAN ASTRONOMY IS ABOUT TELESCOPES
quickmeme.com

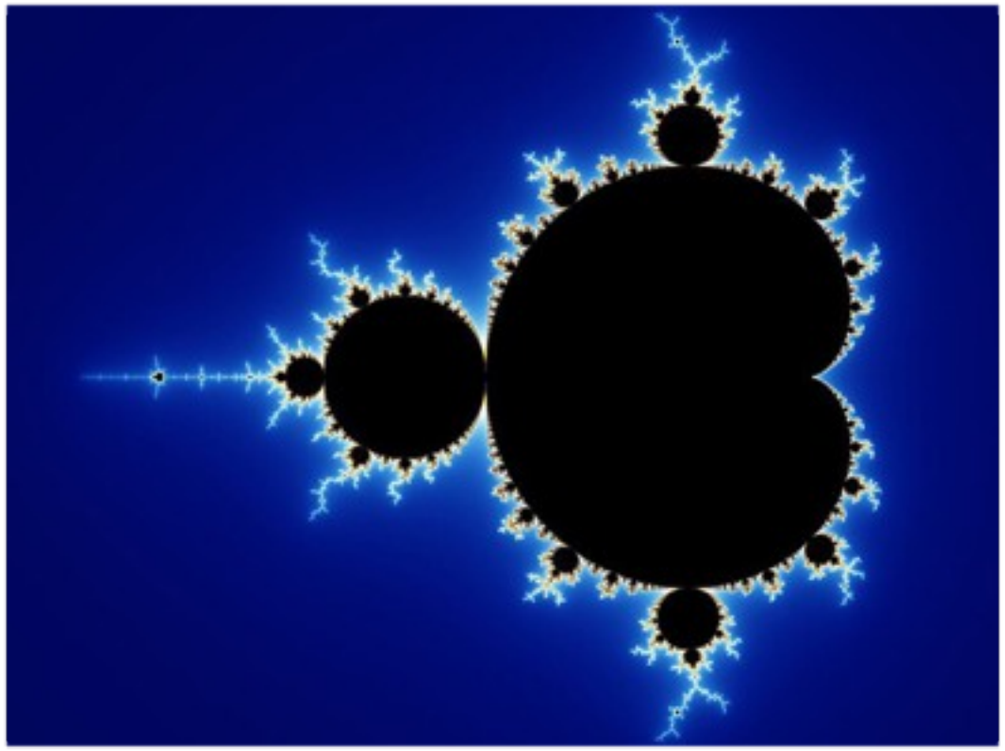
Complexity Management

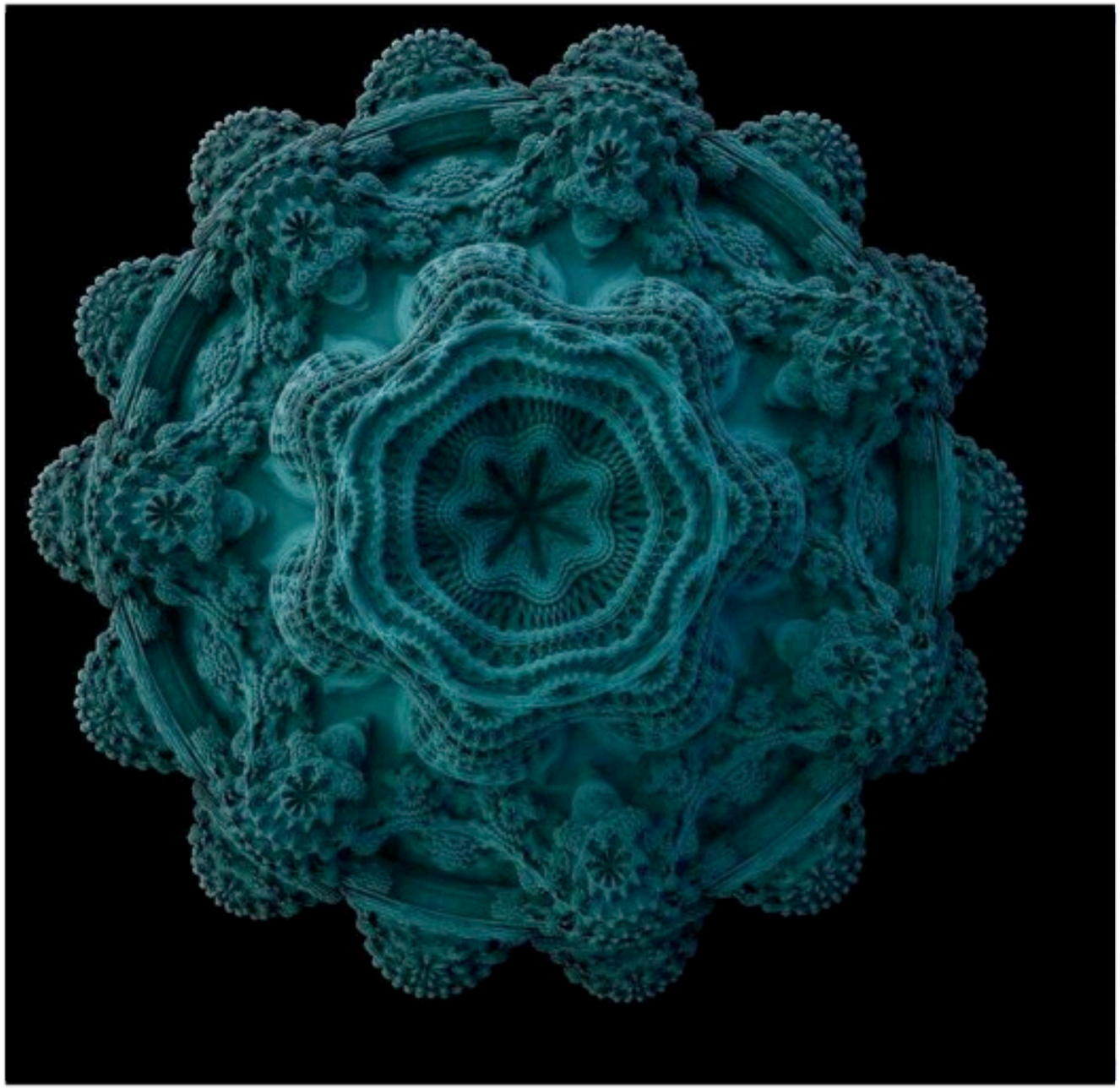
- Software engineering is the process of creating and deploying a system with a specified usage
- A system is a collection of interrelated components, working together to achieve a goal
- Components interact with each other to form the overall system behaviour
- You cannot always predict the behaviour of a system!

Complex Adaptive Systems

- Systems which produce behaviours arising due to the interaction between components
- Non-linear systems
- Interactions can produce synergistic effects, or can result in wildly unpredictable behaviour

Emergence!



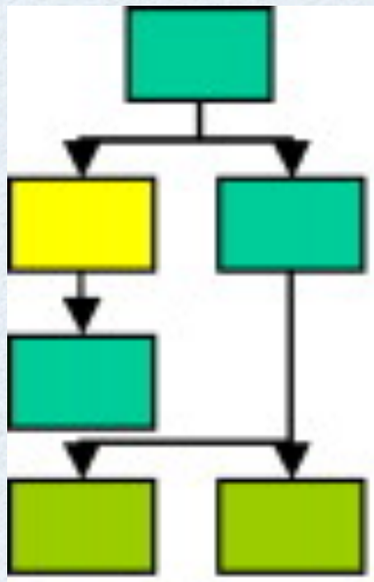


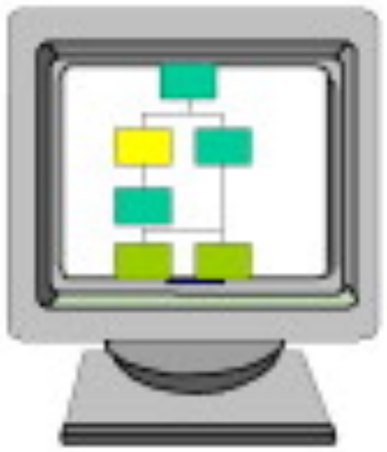
Types of system

- A system can be either technical or socio-technical
- Technical systems:
  - a system which operates without human input
- Socio-technical systems
  - involve the intervention of users, organisations and policies
- *Can you think of examples of each?*

Socio-technical system features

- They have emergent properties
  - properties of the system of a whole that depend on the system components and their relationships.
- They are non-deterministic
- They are interlinked with institutional policies
- The extent to which the system supports the aims is dependent on both the system itself and its users



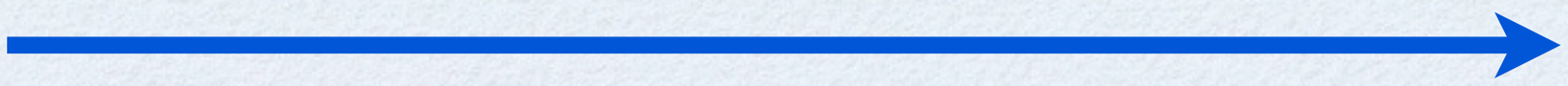


1. Algorithms

2. Computer Programs

3. Human Computer Interaction

4. Socio-technical systems



complexity

binary search

ubuntu

photoshop

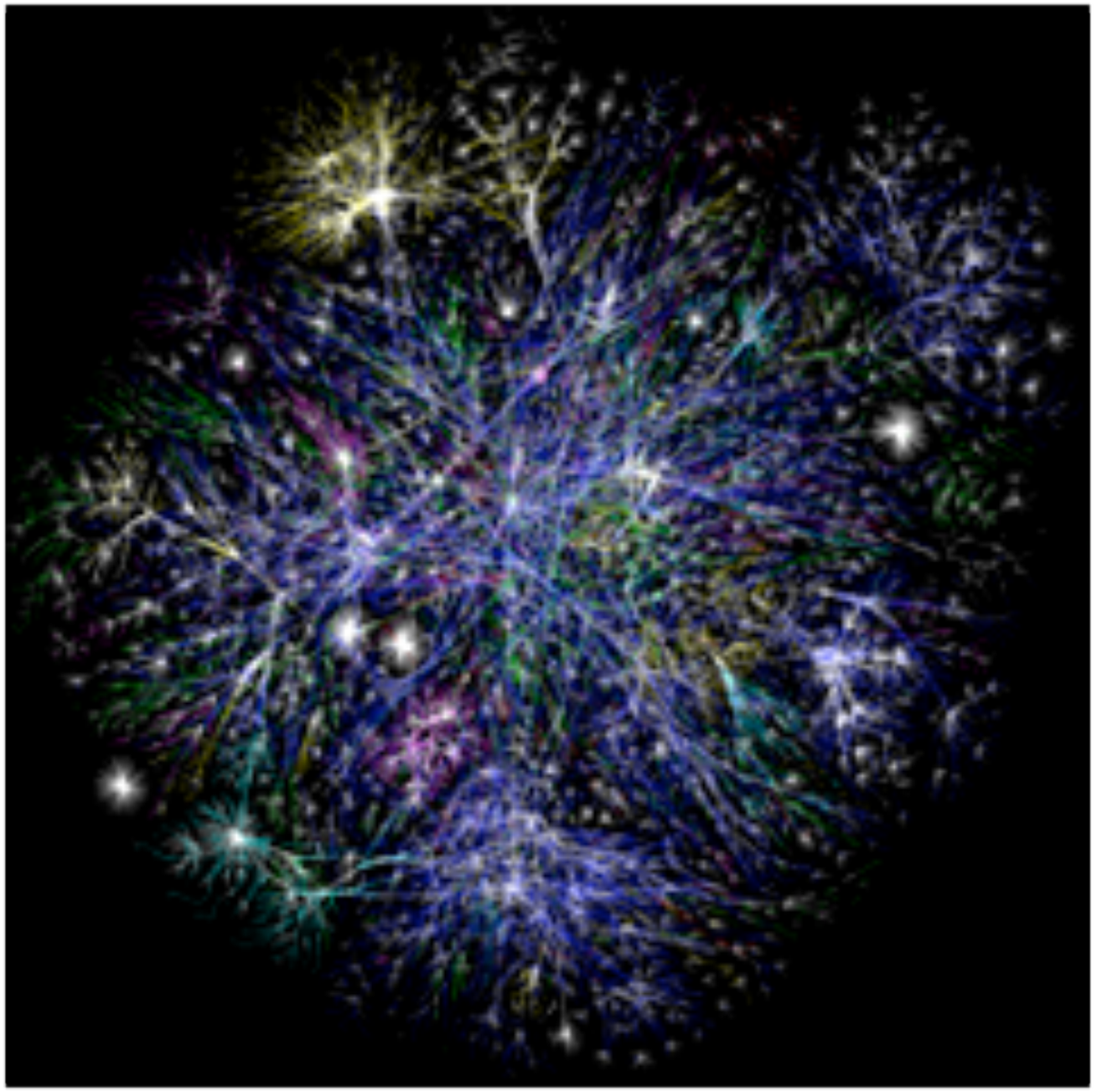
moodle

Constant battle with complexity

- Properties of the system as a whole rather than properties that can be derived from the properties of components of a system
- They can only be assessed and measured once the components have been integrated into a system
- Components can work together to produce good properties - synergy
- This can turn bad producing unpredictable effects

Systems can be unreliable

- Because of component inter-dependencies, faults can be propagated through the system.
- System failures often occur because of unforeseen inter-relationships between components.
- It is probably impossible to anticipate all possible component relationships.
- Software reliability measures may give a false picture of the system reliability.



Manage complexity by adhering to a common set of methods

Defining a software process

- A set of activities whose goal is the development or evolution of software.
- Generic activities in all software processes are:
  - **Specification** - what the system should do and its development constraints
  - **Development** - production of the software system
  - **Validation** - checking that the software is what the customer wants
  - **Evolution** - changing code in response to demands

Ideally a linear process?

Requirements analysis

Software design

Implementation

Integration

Testing

Maintenance

A software process model (SPM)

- A simplified representation of a software process
  - Workflow perspective - sequence of activities;
  - Data-flow perspective - information flow;
  - Role/action perspective - who does what.
- Generic process models e.g.:
  - Waterfall;
  - Iterative development;
  - Component-based software engineering.

What makes good software?

- The software should deliver the required functionality and performance to the user:
  - Maintainability: software must evolve to meet changing needs
  - Dependability: software must be *trustworthy*
  - Efficiency: software should not make wasteful use of system resources
  - Acceptability: software must accepted by its users and be understandable, usable and compatible with related systems

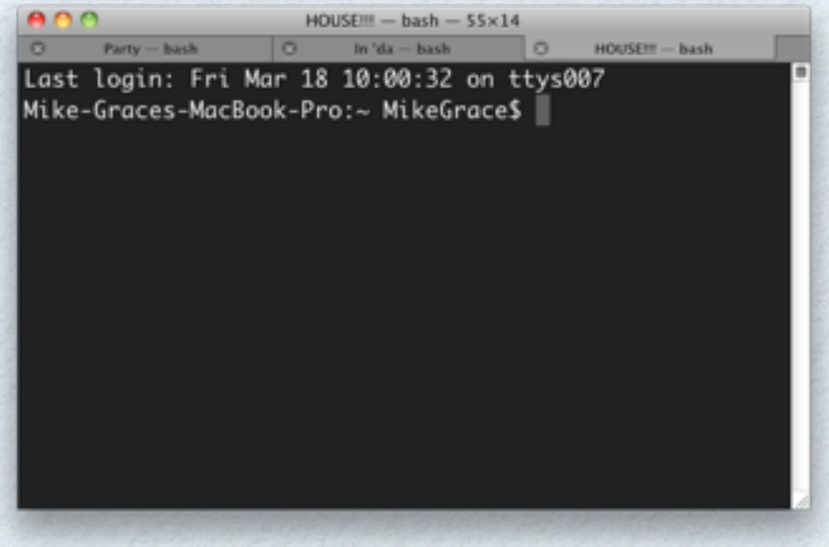
How much is the practical cost?

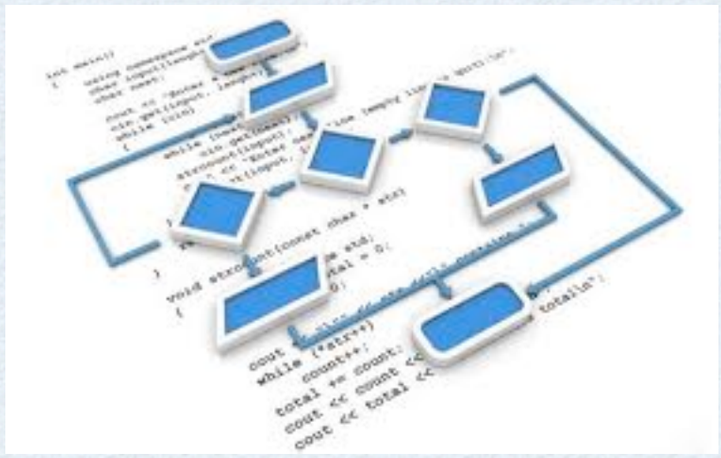
- Varies in proportion to the complexity of the project
- Roughly 60% of costs are development costs, 40% are testing costs
  - for custom software, evolution costs often exceed development costs.
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- Distribution of costs depends on the development model that is used.

Methodology Overview

Order in chaos

- Manage the complexity of interactions between systems of systems
- Unified development process can reduce complexity
- To place program development in context within a larger organisational structure
- Maintain standards


HOUSE!!! — bash — 55×14
Party — bash
In 'da — bash
HOUSE!!! — bash
Last login: Fri Mar 18 10:00:32 on ttys007
Mike-Graces-MacBook-Pro:~ MikeGrace$




PROGRAM
DEVELOPMENT
SOFTWARE AGREEMENT
TROUBLESHOOTING
SCENE
ALTERNATIVES
INTEROPERABILITY
VARIETY
PARADIGM
FORMAL
SOFTWARE
FOUNDATION
MANAGERS
CONVEY
CHANGE
IMPORTANT
GOOD
DOCUMENTATION
OBTUSE
COMMUNICATE
COMMON
PREFERENCE
ROUTINE
CONSTRAINTS
DERIVATIVE

Method != Methodology

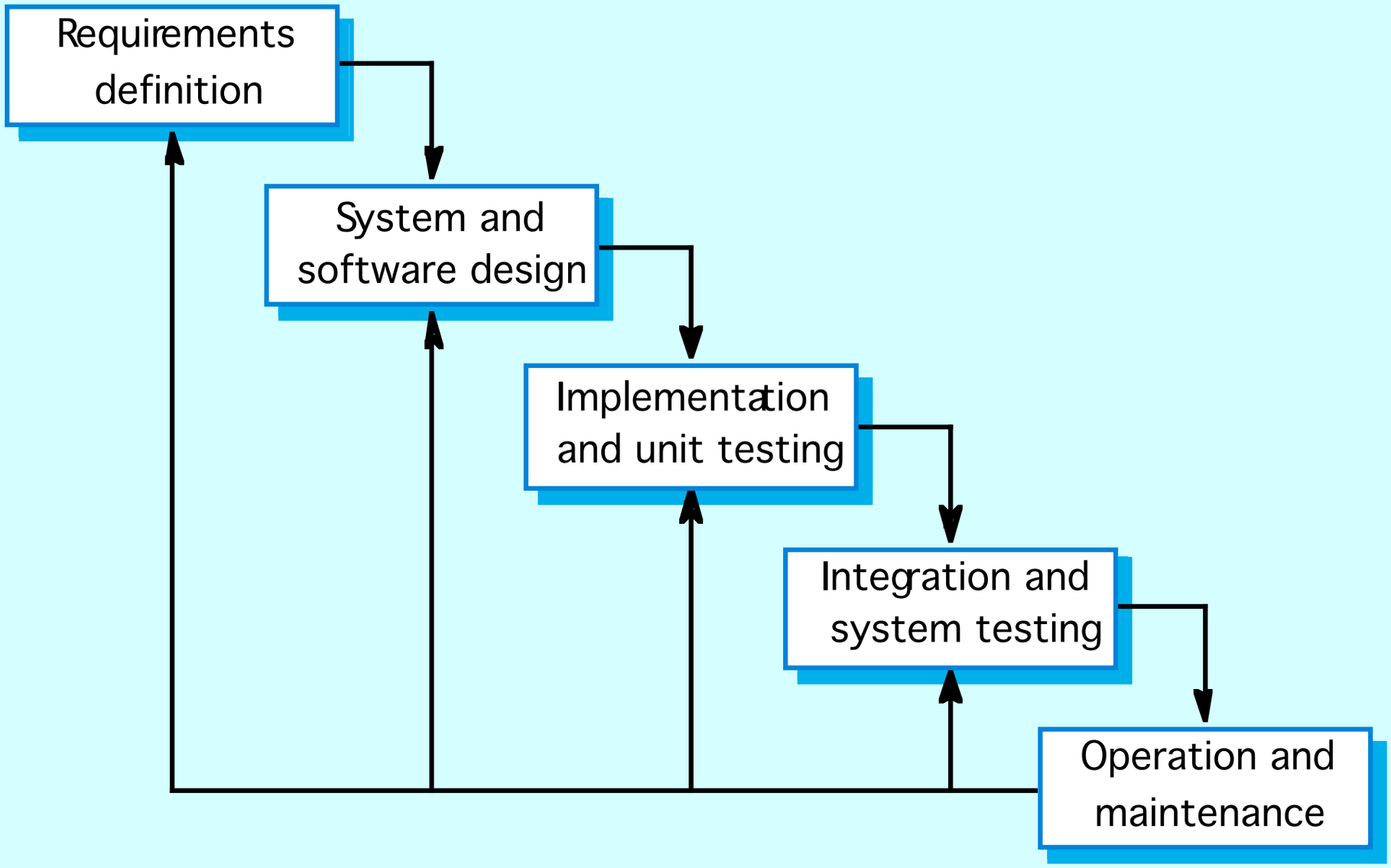
- A methodology is " *a set or system of methods, principles, and rules for regulating a given discipline, as in the arts or sciences*"
- A software engineering methodology is a set or system of methods, principles, and rules for regulating the process of creating and deploying software
- Techniques which are the methodologies are also called software process models

Generic Software Process Models

- The waterfall model
  - Separate and distinct phases of specification and development.
- Iterative/Evolutionary development
  - Specification, development and validation are interleaved.
- Component-based software engineering
  - Systems assembled from pre-existing components

Waterfall Model

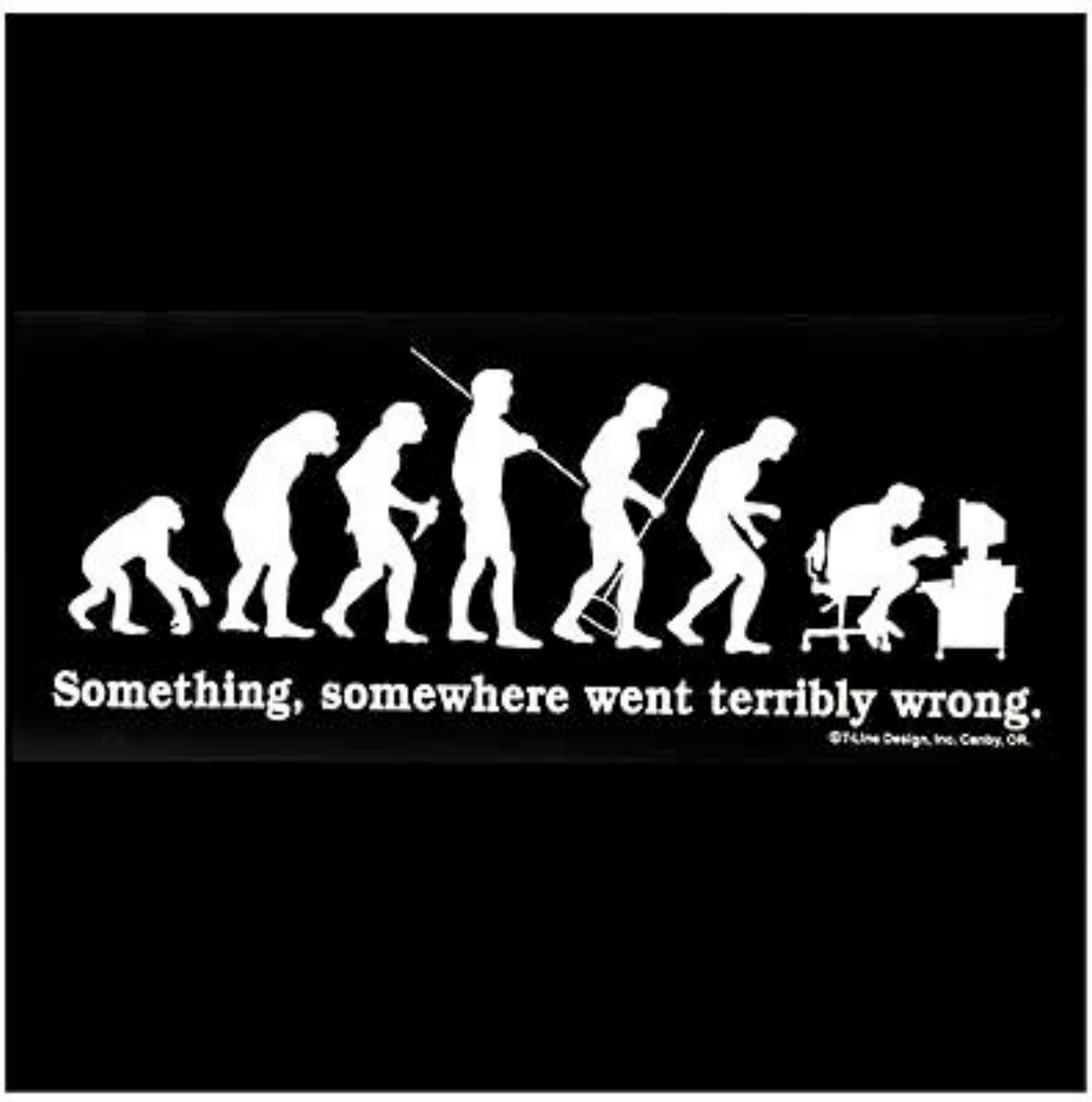
The oldest trick in the book



Many problems!

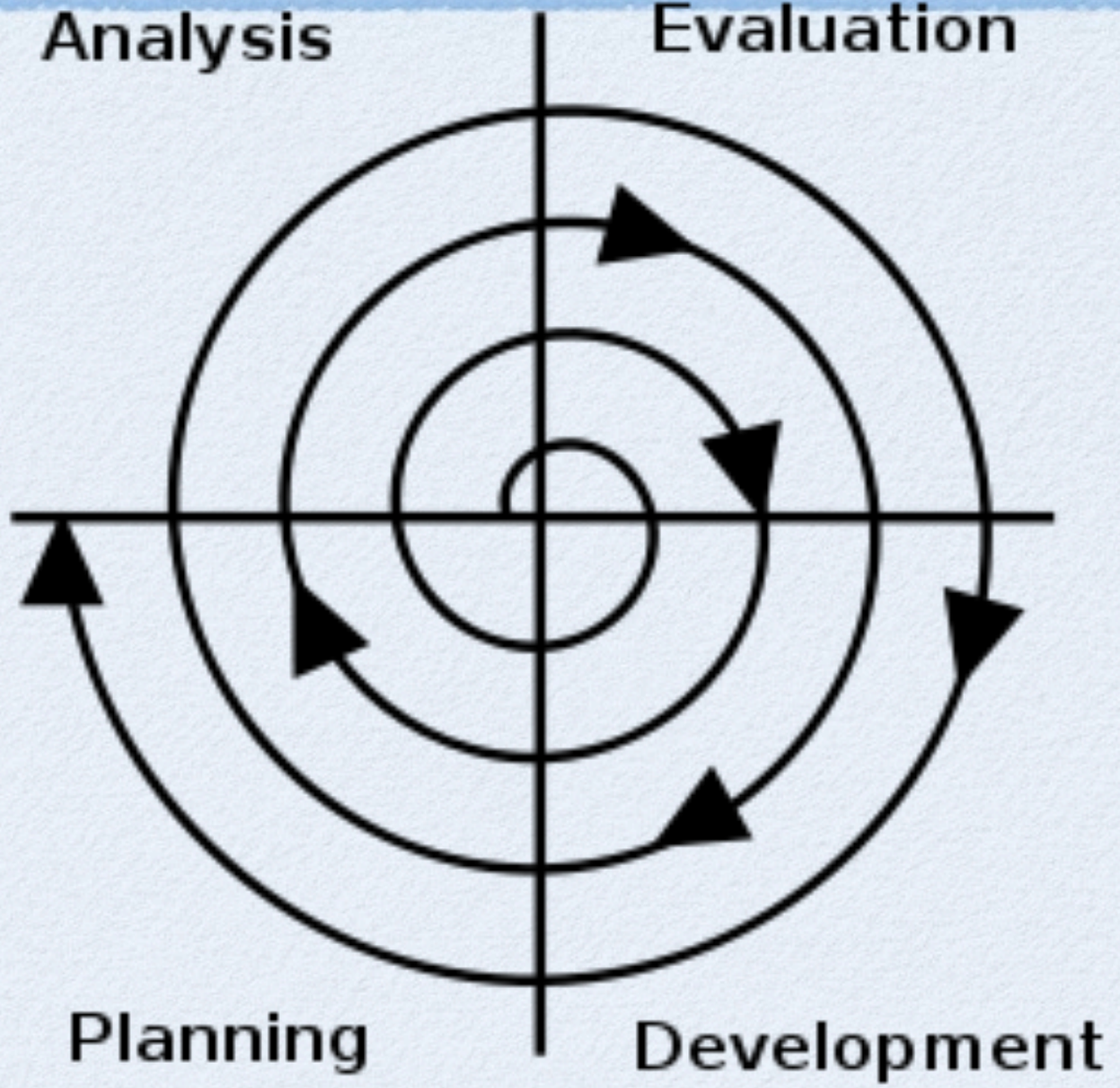
- Needs stable and "perfect" requirements
  - cant always anticipate what you are going to have to do
- Does not account for revision or refactoring
- Too inflexible and static
- Depends on getting each stage exactly right
  - changes can have untoward knock-on effects

Iterative and Evolutionary



Prototyping approaches

Spiral Model



Agile Methods

- A bottom up approach
- Rapid prototyping
- Iterative design
- Based on five principles
- Includes Xtreme Programming

Criticisms of iterative approaches

- Lack of process visibility
- Systems are often poorly structured
  - too much bottom up and not enough planning
- Special skills (e.g. in languages for rapid prototyping) may be required
- Lightweight documentation taken to mean no documentation

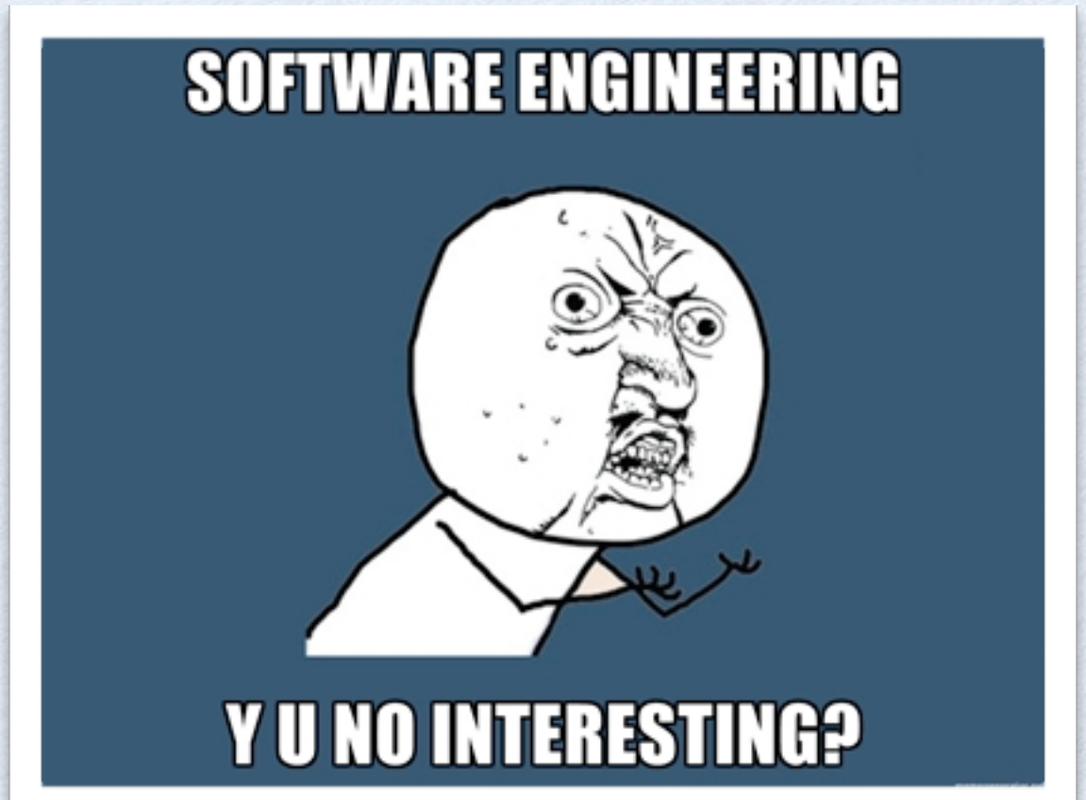
Component-based

- Modular design build on verified existing code
- Linux kernel and UNIX philosophy built on this
- Spend time explicitly testing the interactions between components
- Builds on a platform of previously tested code

And Many More.....

- Behaviour Driven Development
- The Boehm-Spiral Methodology
- The Booch Methodology
- Object Modeling Technique (OMT)
- Rational Objectory Methodology
- WinWin Spiral Methodology
- Test Driven Development

Will cover each one of these methods in detail



...and how to apply them

What we looked at

- Frequently asked questions
- Emergence in software engineering
  - coping with the development of complex systems
- Methodology roundup & process models
  - Waterfall, iterative and component based

Next time...

- Tuesday lab session
- Thursday: Software requirements
- Friday: Writing SRS documents