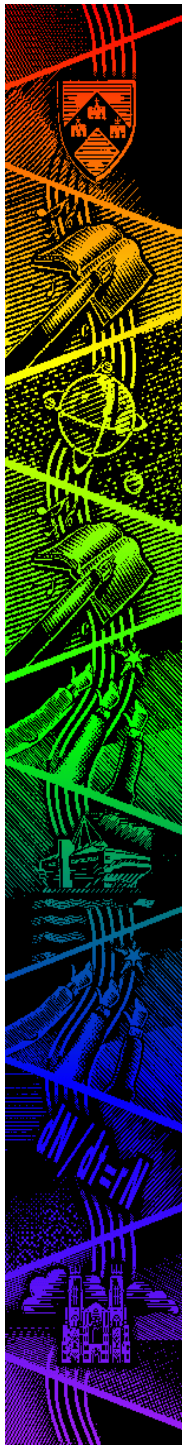
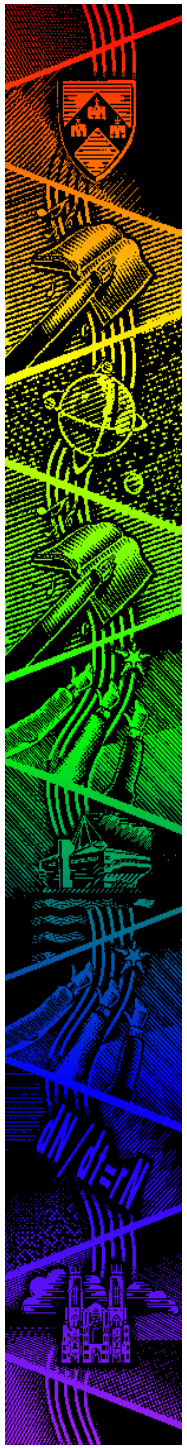


Empiricism, realism and rationalism

A summary of the philosophy

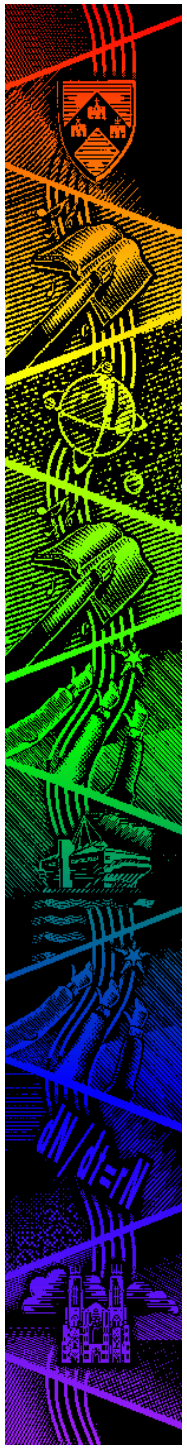
THE UNIVERSITY of York

Chris Kimble
January 2008



Overview

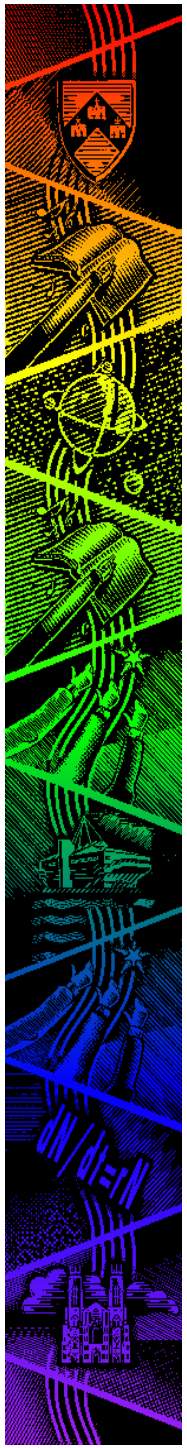
- An end to Philosophy
  - Reality, Representations and Descriptions revisited
- An end to history
  - Philosophy and the history software design
  - Philosophy and the history change
- An end to (my) conclusions
  - The shape of things to come



Review

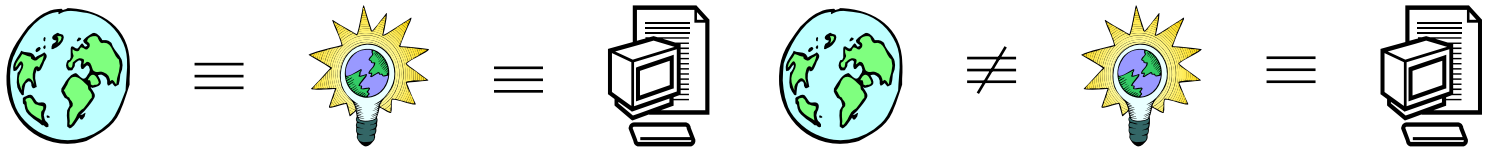
Last week …

- Philosophy
  - Reality, Representations and Descriptions
- Early and late period in software development
  - Earlier period
    - Creation of a valid software description is the problem
    - Creation of correct code is assumed to be OK
  - Later period
    - Creation of valid description assumed to be OK
    - Verifying that code is correct is a problem
- A methodology = a collection of tools, techniques and methods unified by a common philosophy



Review

- Using our notion of equivalence from lecture 1, we now have four ways to think about the problem

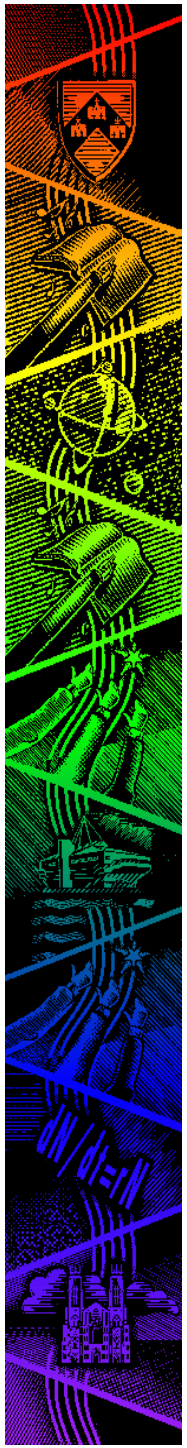


Programs

Old approaches

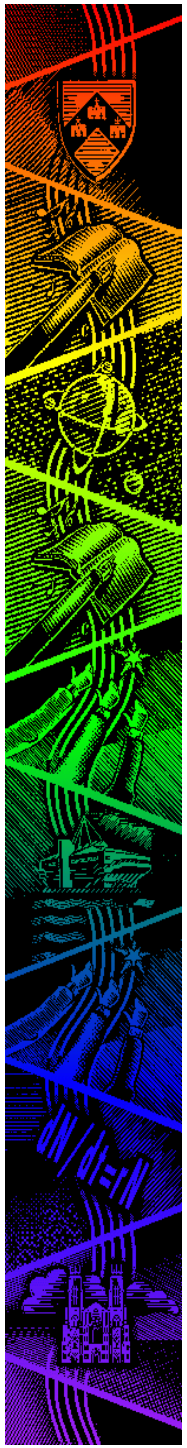
New approaches

Others?



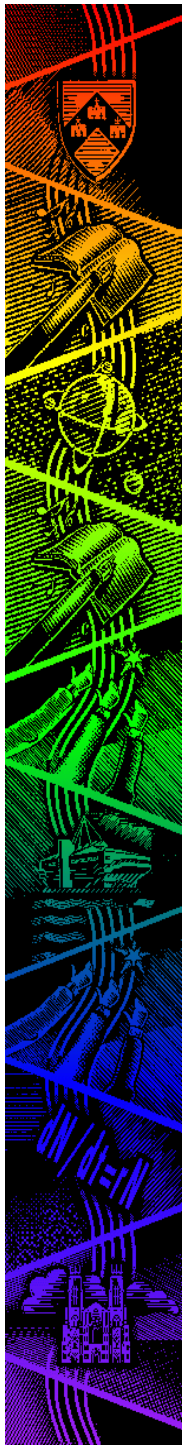
Some questions

- Do any of the previous categories actually exist in practice?
  - Can we find examples of each, and what are they?
- Is there any theoretical explanations for these categories?
  - What is the explanations?
  - What are their implications?
  - How well do the examples match the theory?



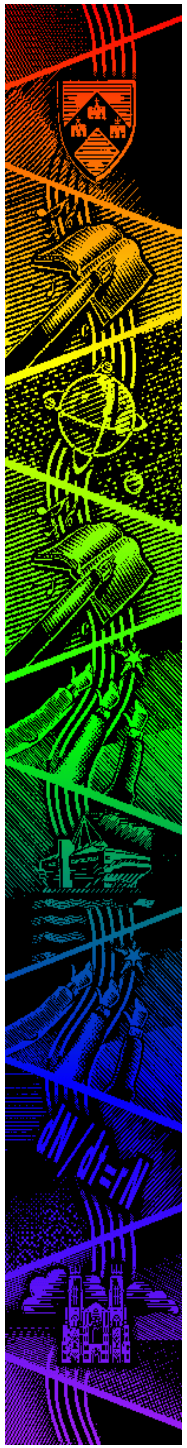
Some assertions

- *Formal Software Design Methods*
  - Examples: Unity, Z, VDM
  - Assumes software and program descriptions to be equivalent (i.e. both are complete and closed)
  - By removing any distinction between software and programs, the formal strand seeks to introduce mathematical rigour into both program and software design



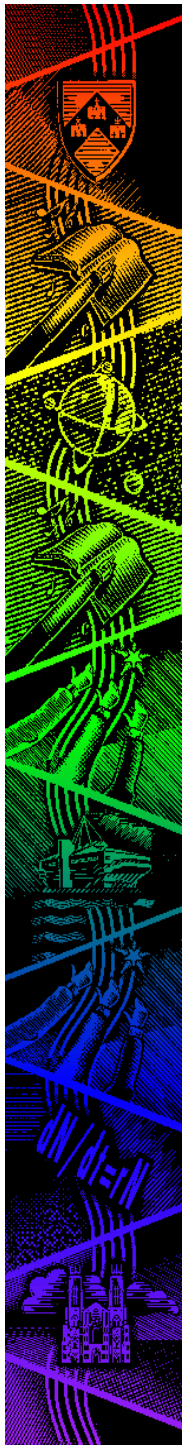
Some assertions

- *Semi formal Software Design Methods*
  - Examples: Jackson System Development, Structured Systems Analysis and Design Method
  - Has strong links to program design methods
  - Assumes the software description is complete, but not closed
  - The focus is on the logical flow of control in the program, which frees the programmer from having to be concerned with any physical details of the implementation



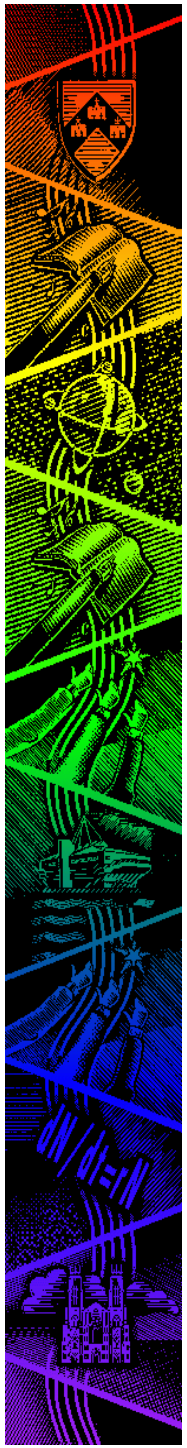
Some assertions

- *Object orientated Software Design Methods*
  - Examples: Booch Object Oriented Design, Rational Unified Process
  - Has strong links to object oriented languages
  - Assumes the software description is closed but not complete
  - Uses (closed) reality as a baseline to free the designer from concerns about the problems of dealing with inaccurate representations of the physical system



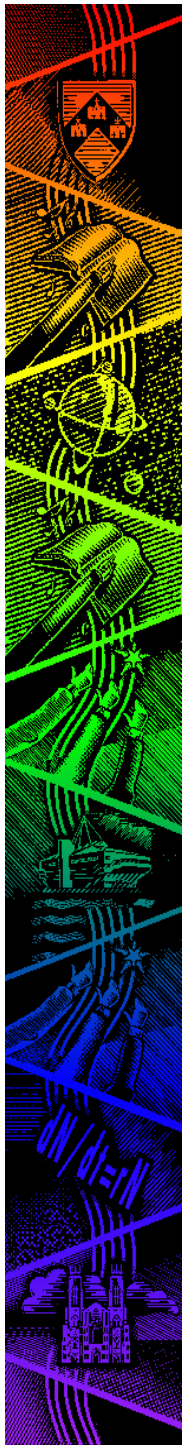
Some assertions

- *Holistic Software Design Methods*
  - Examples: Soft Systems Methodology, Ethics
  - Attempts to look at the whole system
  - Abandons any relationship between program design and software design
  - Does not assume the software description is closed or complete



Underlying Theory

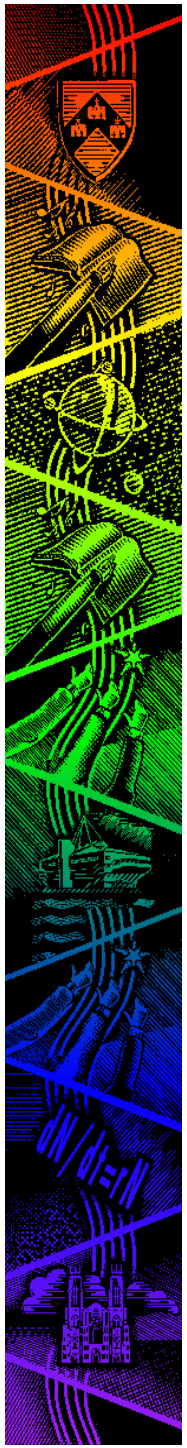
- We have asserted that there are examples of methods that to fit into each of the four categories
- However, if a methodology is “a collection of tools, techniques and methods unified by a common philosophy”, what is the philosophy that underlies each?
- Is there any theoretical basis for this classification?



The last slice of philosophy

- Epistemology (Rationalism and Empiricism)
  - Epistemology is concerned with theories of knowledge, asking: “What can we know and how do we know it”? Epistemological arguments are those that focus on the study and characterisation of knowledge.

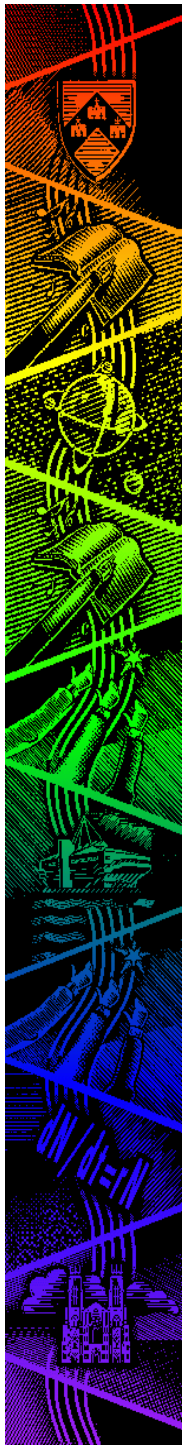
- Ontology (Realism and Anti Realism)
  - Ontology is concerned with theories of existence, asking: “What is the essence and nature of the world”? Ontological arguments are concerned with the nature of reality; without any concern for how that nature might be ‘known’.



Rationalism and Realism

- *Rationalist* arguments deal principally with epistemology claiming that reason is source of all knowledge and that everything that can be known, must be intelligible and rationally explicable

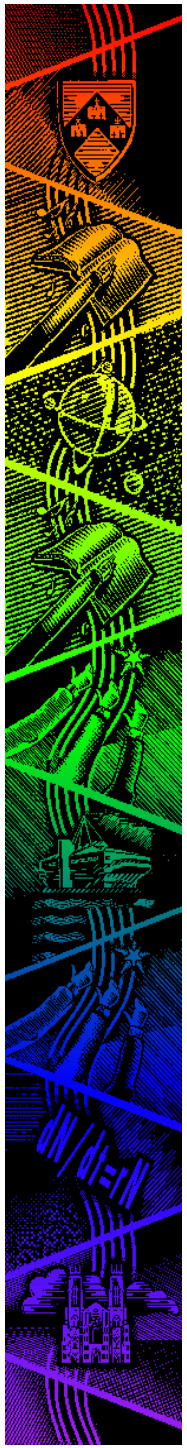
- *Realist* arguments deal principally with ontology claiming that there is such a thing as truth and that all beliefs can be tested against a reality that is knowable



Empiricism and Anti-Realism

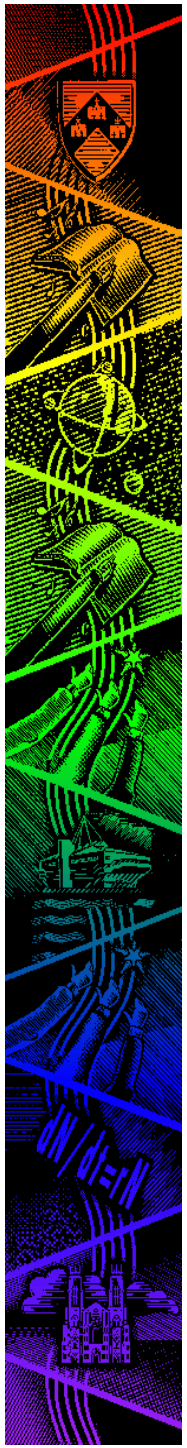
- *Empiricist* arguments deal principally with epistemology claiming that all knowledge derives from observation. Everything that can be known, can only be known through experience.

- *Anti-Realist* arguments deal principally with ontology claiming that the perception of reality is so bound to the mind that observes it, that it is impossible to conceive of the 'true' nature of objects



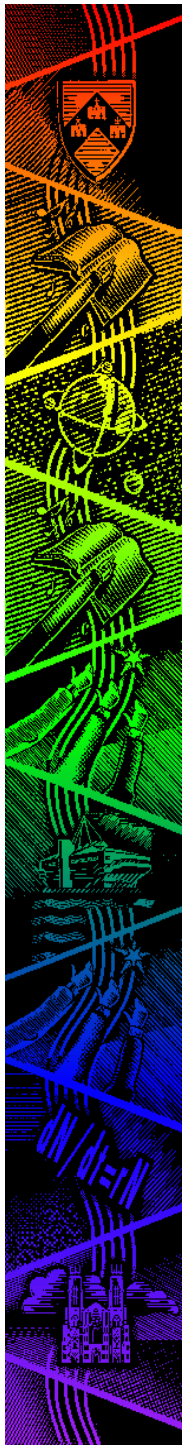
Philosophy and Software Design Methods

- *Formal*
  - Examples: Unity, Z, VDM
  - There is a seamless equivalence between the software description, the representation in the designers mind and the underlying aspects of reality that are being modelled
  - There is an answer that is ‘true’ and this can be discovered by the application of logic and reason = realist ontology and rationalist epistemology



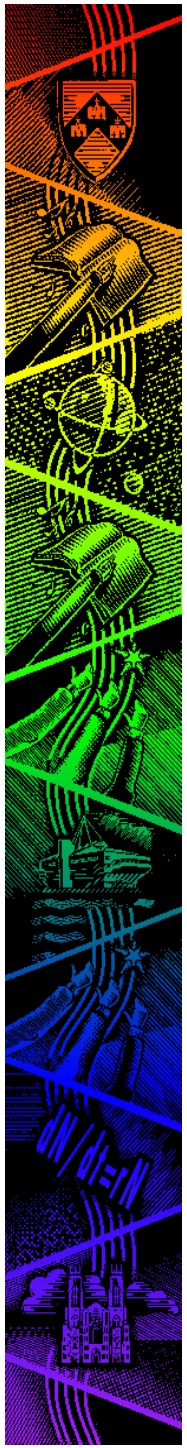
Philosophy and Software Design Methods

- *Semi formal*
  - Examples: Jackson System Development, Structured Systems Analysis and Design Method
  - Software designs that are logically correct do not always reflect the properties the same software design has in reality
  - Allowing a split between logical and physical designs = anti-realist ontology and rationalist epistemology



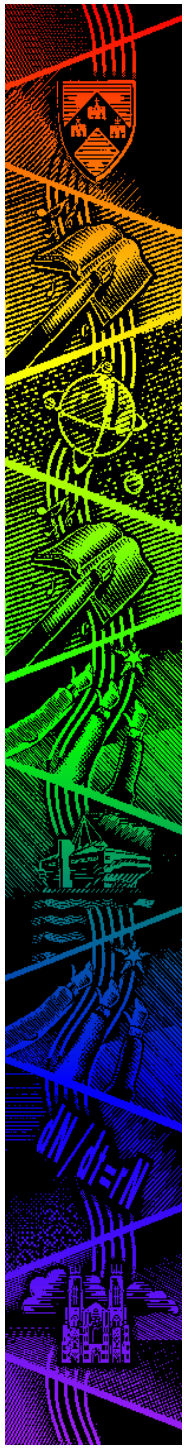
Philosophy and Software Design Methods

- *Object orientated*
  - Examples: Booch Object Oriented Design, Rational Unified Process
  - The software description is formed from observation of reality
  - Rather than working from the description to reality, these methods work from reality back to the description = realist ontology and empiricist epistemology



Philosophy and Software Design Methods

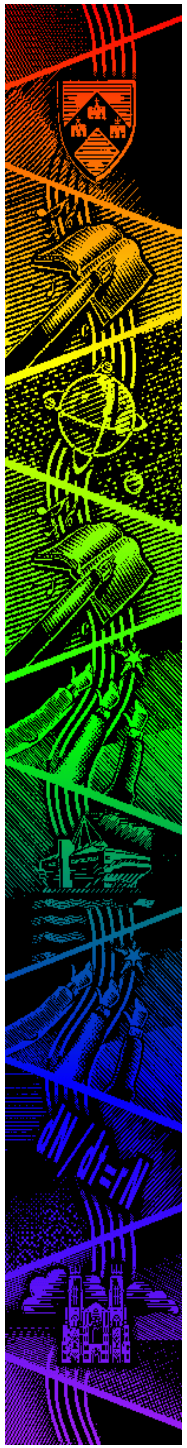
- *Holistic*
  - Examples: Soft Systems Methodology, Ethics
  - Relationships between features of the software design and reality are always a matter of conjecture and open to challenge
  - Anti-realist ontology and empiricist epistemology



Philosophy and Software Design Methods: a Summary

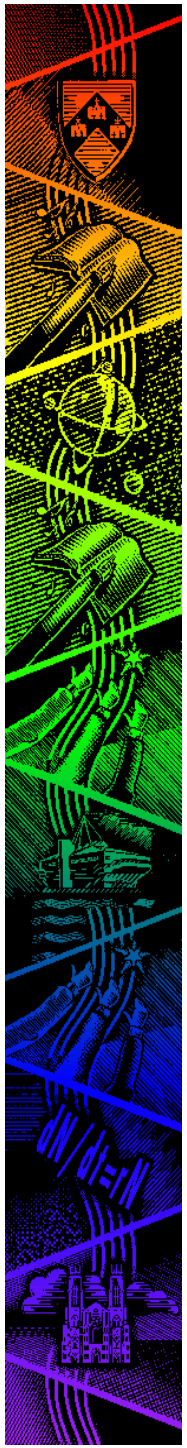
Research Strand	Ontological Position	Epistemological Position
Formal	Realist	Rationalist
Semi-Formal	Anti-Realist	Rationalist
Object-Oriented	Realist	Empiricist
Holistic	Anti-Realist	Empiricist

Chris Kimble
January 2008

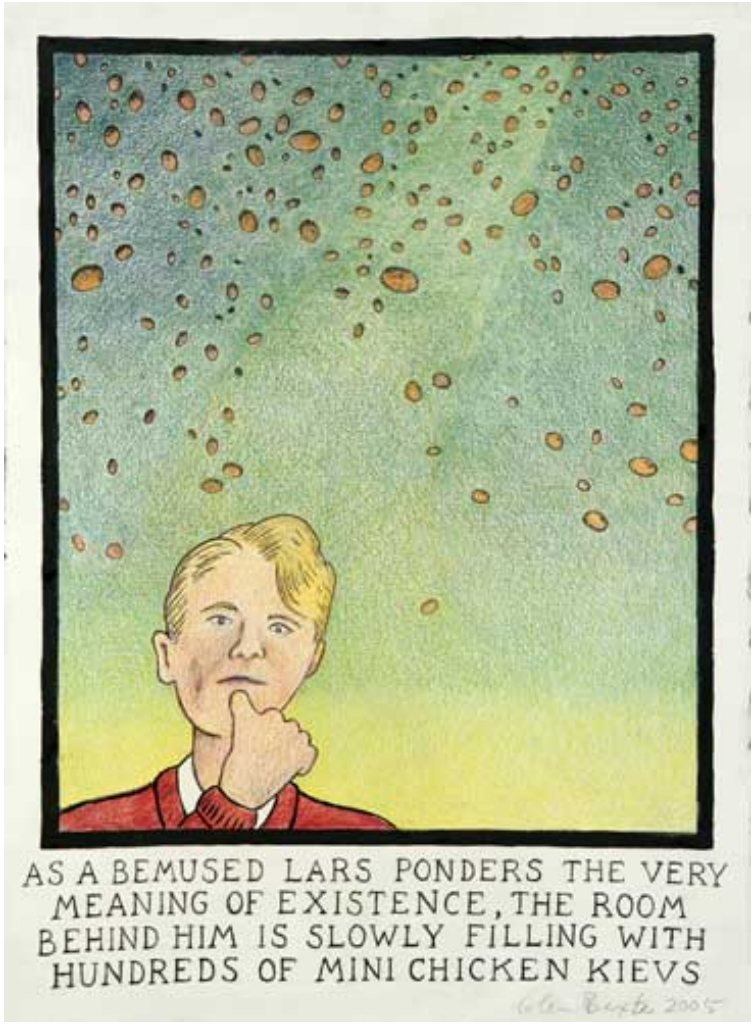


Review

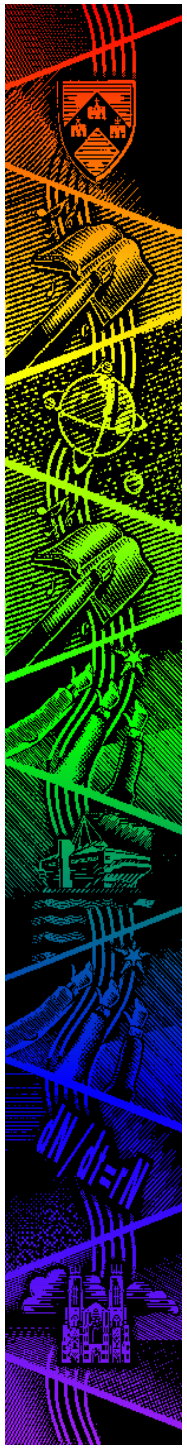
- We have:
  - A way to describe equivalence
  - An understanding of the differences between programs and software
  - An understanding of the different approaches that *might* be taken to developing software
  - A classification of software design methods that can be shown to have both “real world” examples and a basis in theory
- Is that all we need to think about?



Exercise

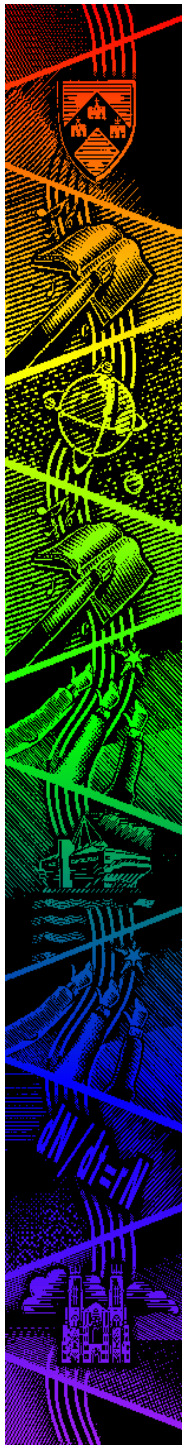


- Review the material we have covered so far
- What are the practical implications of our theoretical contemplations?



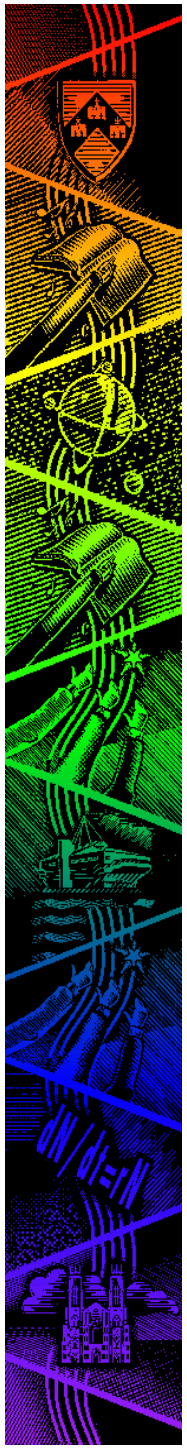
Implications of theoretical standpoint

- Descriptions and Representations
  - The distinction between descriptions and representations deal with the knowledge relationship and hence are the concern of epistemology
- Representations and Reality
  - The distinction between representations and reality deal with the information relationship and hence are the concern of ontology



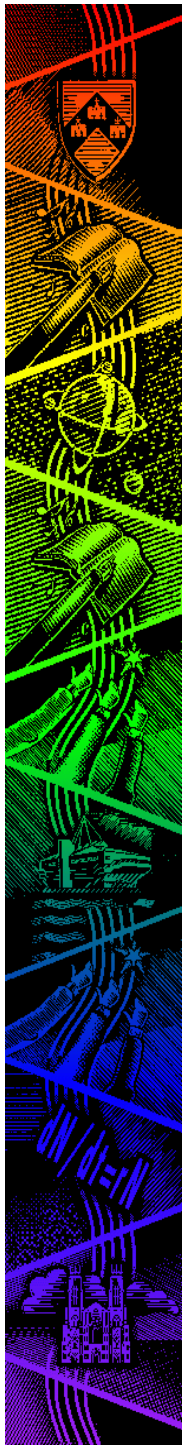
Descriptions and Representations (Epistemology)

- Implications of an empiricist viewpoint
  - The designer forms their *representation* of reality, and then from this experience, they form a *description* of their representation.
  - However, forming the description is an experience and so the representation will change
  - The representation and the description are always out of step and the description is always incomplete
- Consequences
  - Empiricist arguments offer a natural way to deal with change as a result of experience
  - Because different designers can have different representations design consistency can be a problem



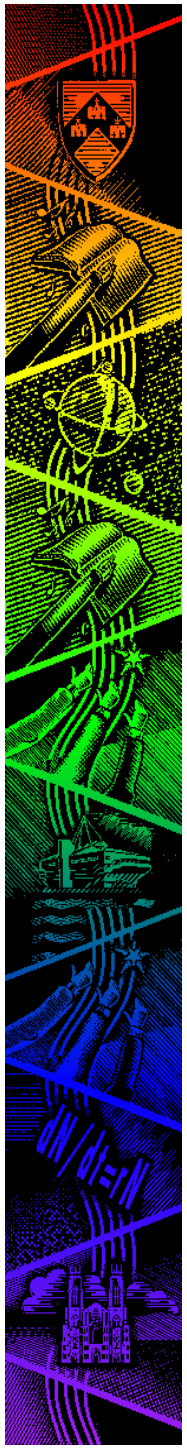
Descriptions and Representations (Epistemology)

- Implications of an rationalist viewpoint
  - The designer starts with the *description* and, by applying the principles of reason, forms a *representation* from the description.
  - Because reason is independent of experience, the process now stops
  - The description and representation remain in step and complete - even if the description changes.
- Consequences
  - Rationalist approaches produce static software descriptions with no means of dealing with change
  - By equating the description with the representation, they lose any notion of designer experience



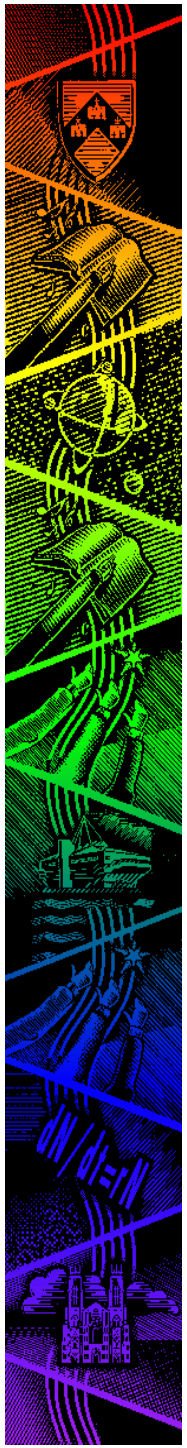
Representations and Reality (Ontology)

- Implications of an Realist viewpoint
  - Realist arguments emphasise that reality is independent of a designer
  - Because there is only one ‘true’ reality it each designer’s representation is based on the same reality
  - All of the representations and the underlying reality remain in step even if the underlying reality changes
- Consequences
  - As each designer is working from the same reality, realist arguments are likely to help software designers produce software that is more cohesive
  - realist arguments deny the validity of different viewpoints and may produce designs that lack innovation

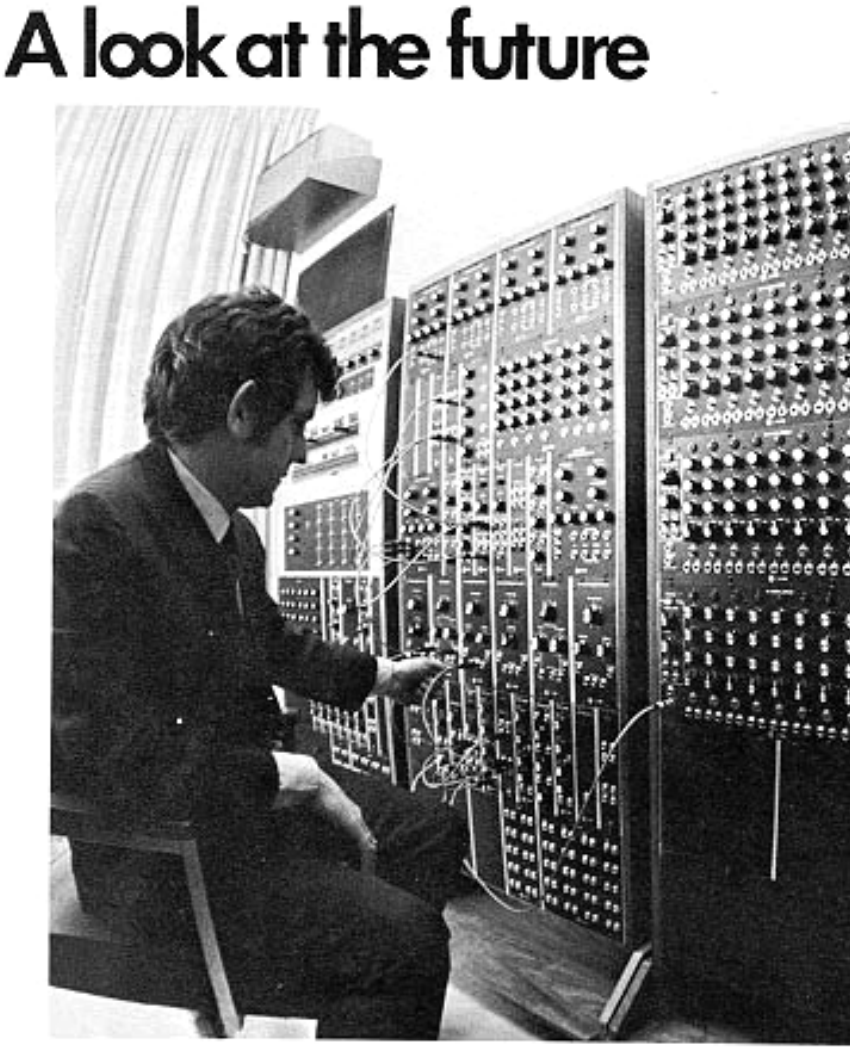


Representations and Reality (Ontology)

- Implications of an anti-realist viewpoint
  - The representation of an individual designer is unique to that designer
  - Reality only exist to the extent that is perceived by an individual
  - Representations and reality are never in step
- Consequences
  - As each designer is working from a different reality, anti-realist arguments mean that it is unlikely that software designers will produce cohesive software
  - Anti-realist arguments deny the designer the opportunity of establishing common ground with either clients of other designers



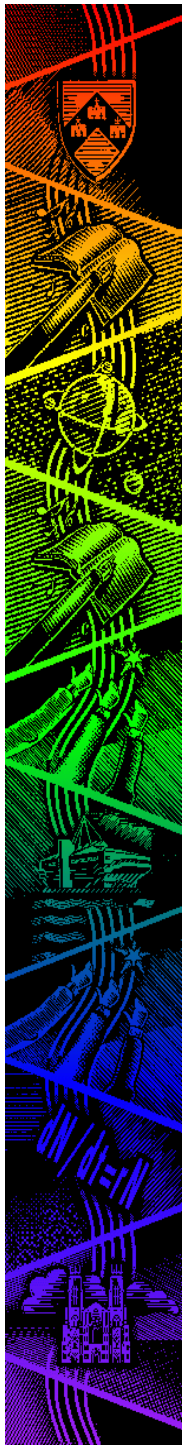
What's next?



- ✓ Do any of the previous categories actually exist in practice?
  - ✓ Can we find examples of each, and what are they?
- ✓ Are there any theoretical explanations for these categories?
  - ✓ What is the explanations?
  - ✓ What are their implications?
  - × How well do the examples match the theory?

THE UNIVERSITY of York

Chris Kimble
January 2008



The shape of things to come

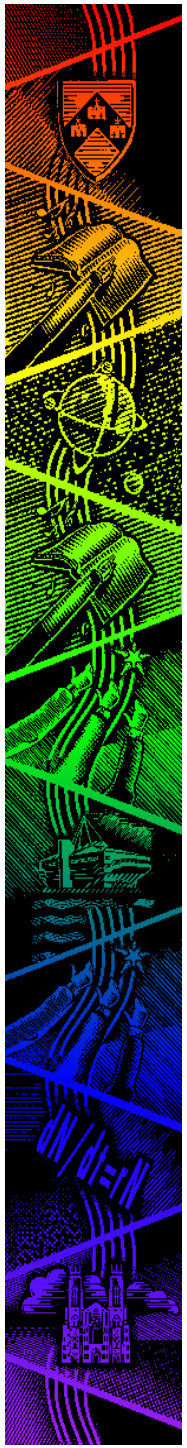
- Next week we begin to look at specific examples
- The week after, practical sessions begin
- I give a lecture one week, you present a seminar paper on the same topic the next week which answers the question
  - "How well do the examples match the theory?"

- Week 4

  (me) Formal Methodologies

  - (e.g. Unity, Z, VDM)

  (you) Background reading (in own time)



The shape of things to come

- Week 5

  (me) Semi-Formal or Structured Methodologies

  – (e.g. Jackson System Development, Structured Systems Analysis and Design Method)

  (you) Background reading (in own time)

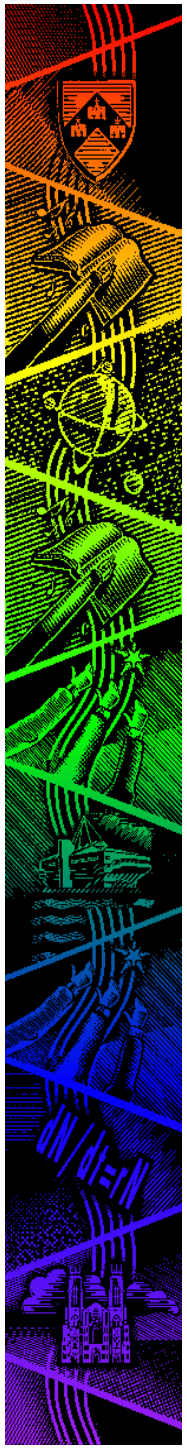
  (you) Formal Methodologies (in practical slot)

- Week 6

  (me) Object-Oriented Methodologies

  – (e.g. Object Process Methodology, Rational Unified Process, Object Modelling Technique)

  …



The shape of things to come

- Week 7

  (me) Holistic Methodologies

  – (e.g. Soft Systems Methodology, Ethics)

  …

- Week 8

  (me) Blended /Mixed Methodologies

  – (e.g. Merise, Multiview)

  ….

- Week 9

  (me) Review of everything

  (you) Blended /Mixed Methodologies (in practical slot)