



THE AUSTRALIAN NATIONAL UNIVERSITY

It's not Science, it's ICT

Shirley Gregor

Australasian Conference on Information Systems
Toowoomba, 6 December, 2007

1. Which paradigm?
2. Problems of 'scientism'
3. Answer A: Different types of theory
4. Answer B: Design theory
5. Questions for further work

1. Which paradigm?



Knowledge Area Examples



Science Paradigm	Applied Paradigm (e.g. Information Systems)
<ul style="list-style-type: none"> ▪ Gas Laws 	<ul style="list-style-type: none"> ▪ Relational database theory
<ul style="list-style-type: none"> ▪ Theory of relativity 	<ul style="list-style-type: none"> ▪ Systems development methodologies
<ul style="list-style-type: none"> ▪ Learning theories 	<ul style="list-style-type: none"> ▪ Architectural principles of DSS
<ul style="list-style-type: none"> ▪ Institutional theory 	<ul style="list-style-type: none"> ▪ Guidelines for strategic IS management
<ul style="list-style-type: none"> ▪ Cognitive psychology 	<ul style="list-style-type: none"> ▪ IT Governance



Argument

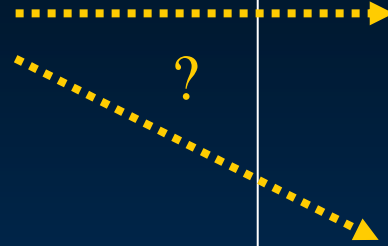
The IS research paradigm should be one of an applied discipline which concerns the design and construction of IT-related artifacts and interventions in the world.

(Other applied disciplines including management, accounting, medicine, architecture, engineering, computer science, economics.)

We should not uncritically adopt our models of research from the science paradigms, either physical or social science.

Interaction between normal science and design science

Normal science theories



Artifact construction

Design theories



Tests of design theories



Examples of work in the design paradigm

- Eric van Heck and colleagues – ‘Making Markets’
- Iversen et al.(2004). Developed a process for managing risks in software development – action research studies (MISQ, 28,3, 395-434)
- Basili et al. - ICIS 2007 governance track – Copyrighted approach for bridging the gap between business strategy and software development
- Peter Weil et al. at MIT – work on IT governance
- Much other work – often not labelled as “design work”

2. Problems of 'scientism' paradigm

- Pre-occupation with idealized views of science
- Pre-occupation with debates about positivism vs interpretivism (largely irrelevant)
- Aim for legitimacy through 'being scientific'
- Grab-bag choice of theory to underpin IS work
- Don't regard design theory as legitimate form of theory (as in "where's the theory?")
- Lack of focus on outcomes and things in the real world that can be manipulated to achieve these outcomes when doing normal science-type work

- Teaching inappropriate research methods (not including ones to do with design approaches)
- Theories/models investigated that do not lead well to design knowledge eg TAM
- Lost opportunity for identifying what really defines ICT disciplines
- Lack of relevance of much IS work to real problems
- Insufficient thinking about what 'design research' really means and how to practice it

3. Answer Part A: Recognize different types of theory



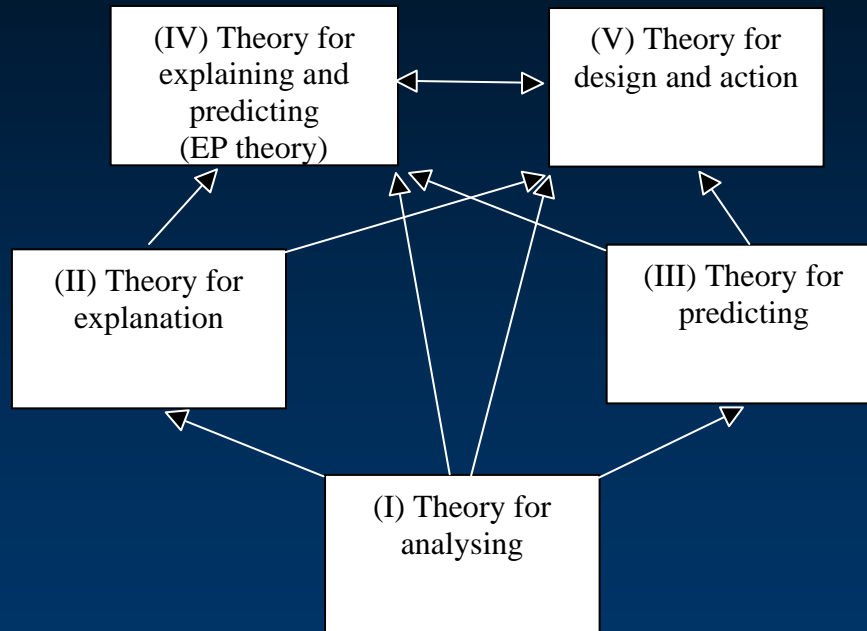
Allowing communication within a common
framework

Types of theory

Different types:

- I – Theory for analysing
- II – Theory for explaining
- III – Theory for predicting
- IV – Theory for explaining and predicting
- V – Theory for design and action

Gregor, S. (2006). The nature of theory in information systems, MISQ, 3,30, 611-642.



Type I – Theory for analyzing

- Says “what is”
- Needed when little known about a phenomenon.
- Can be (a) analysis/descriptive (b) classification (c) construct measurement

Examples:

livari, Hirscheim, Klein (2000) – framework for classifying ISD methodologies

Type II – Theory for explanation

- Says “how”, “why”, “when”, “where”
- Similar to some work in interpretivist paradigm
- Can be:
 - (a) for enlightenment, sensitizing at a high level
 - (b) For explaining particular case(s)

Example:

Orlikowski (1992) – structurational model of technology –
high level of generality

Type III – Theory for predicting

- Says “what will be”
- Can have precise prediction without understanding eg in predicting business activity
- Examples:

Moore’s Law (1965) – relationship between number of computer components per chip and cost over time

Type IV – Theory for explaining and predicting

- Says “what is”, “how”, “why”, “what will be”
- Common view of theory in sciences
- Some may term “positivism” but does not necessarily entail quantitative methods, naïve realism or strong determinism
- Propositions may be probabilistic
- Examples – Weber – Theory of representation

Type V – Theory for design and action

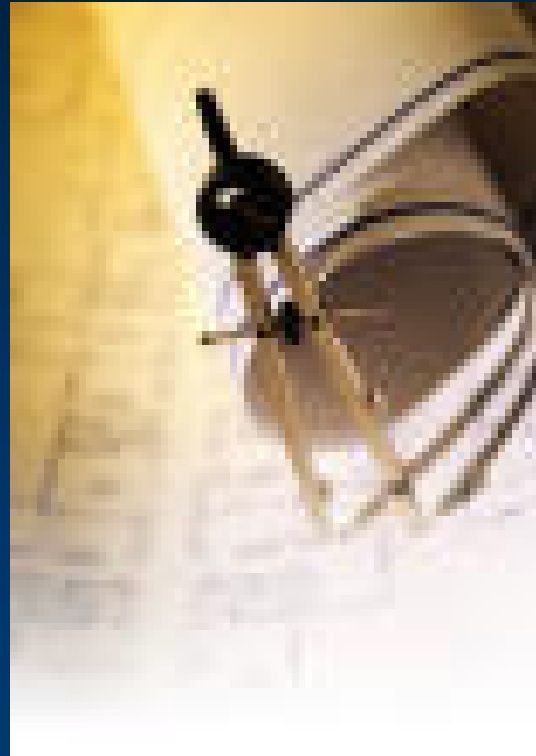
Design is the core of all professional training: engineering, architecture, business, education, law, medicine - and information technology

Need a science of design – intellectually tough, analytic, partly formalizable, partly empirical and teachable.

Herbert Simon - "Sciences of the Artificial"

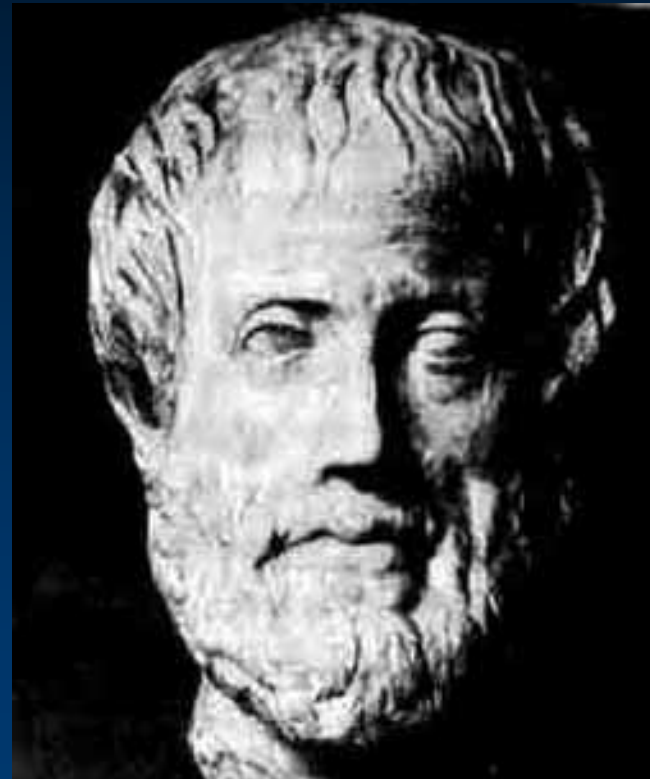


4. Answer Part B: Focus on Design theory



Background?

- Very little on philosophy of technology and design knowledge
- Some ideas on design-type knowledge date back to Aristotle
- Greeks also distinguished between *episteme* and *techne*





Need to know for design knowledge:

1. What an artifact is for (type/class)
 2. The form it takes(features/design principles)
 3. What it is made from (materials)
 4. How it is made (process/designers)
- [Aristotle's Four Causes]

Constructive Research & Design Science

- livari (1991)
- Kasanan et al (1993)
- Nunamaker et al (1991)
- March & Smith (1995)
- Hevner et al (2004)

Tends to focus on the research activity and the artifact (instantiation) as the main product – less on the theory that results

Work in other disciplines

- Architecture (Alexander 1977)
- Management (Van Aken, 2004)
- Management accounting (Kasanan 1993)
- Education (Samuelson 2003)

Seminal work on specifying IS design theory by Walls et al (1992), based on Dubin

But problems –

- Not clear whether design theories are for product or process
- Missed some of Dubin's elements
- Does not address role of instantiations
- May be too complex

Core components

1. Purpose & scope
2. Constructs
3. Principles of form and function
4. Artifact mutability
5. Testable propositions
6. Justificatory knowledge

Additional components

7. Principles of implementation
8. Expository instantiation

5. Questions needing further work

1. Is explanatory theory from the social or physical sciences mandatory for design theory?

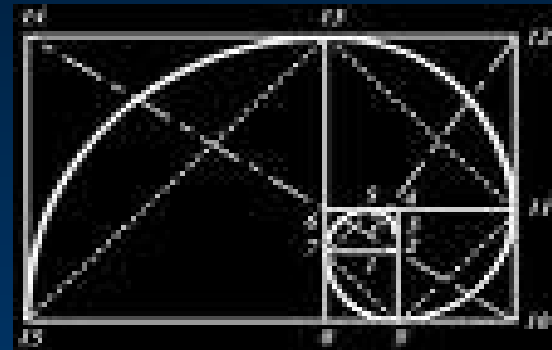
Suggest: Desirable but not essential.

Herbert Simon – artifact construction might precede knowledge of how and why the artifact works

Design theory is linked to theory of other types only where possible.

In some cases supporting explanatory knowledge from other paradigms is little or non-existent

Art Theory



2. Can we show a contribution to design theory just by showing it is an advance on another design theory?

Suggest: Yes – see Iverson risk management approach earlier (in MISQ)

3. Is testing in the normal science tradition necessary or useful for designed artifacts?

Common approach:

Design a feature X (eg explanation facility) to bring about some outcome Y (eg user learning).

Normal science test: Conduct experiment to test proposition "System with X brings about outcome Y"

Problem: Hard to test properly all design decisions in largish systems

4. How far can we go in building generalizations about design?

Suggest: Generalizations will be limited.

Van Aken (2004) in management proposes *technological rules*: “If you want to achieve Y in situation Z, then something like action X will help”

5. How do we abstract theory about the design of artifacts from practice?

Suggest: With great difficulty. What examples do we have?

Project management principles?

IT governance knowledge?

6. What philosophical positions underlie design research?

Suggest: Need some from of realist ontology.

Epistemology?

In designing artifacts (theory building) - critical theory is one approach. Also action research.

In evaluating artifacts – many epistemological positions – multi-method. Can use case studies, interpretive studies, experiments .

Conclusions & questions

