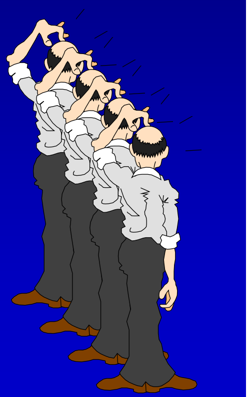
Abstraction Challenges

Panel

Brian Berenbach, Jeff Gray,

Mats Heimdahl, Jeff Kramer



Abstraction important in Software Engineering

Abstraction is fundamental to Engineering in general, and to Software Engineering in particular !

"Once you realize that computing is all about constructing, manipulating, and reasoning about abstractions, it becomes clear that an important prerequisite for writing (good) computer programs is the ability to handle abstractions in a precise manner."

Keith Devlin *CACM* Sept.2003

Panel focus

Modeling's advantages are realized through abstraction mechanisms, such as the ability to model the essential characteristics of an application within the problem space by removing concerns such as platform dependencies that belong in the technical solution space.

However, there is still much work to be done with respect to improving abstraction within modeling languages …

Abstraction?

- the act of withdrawing or removing something
- the act or process of leaving out of consideration one or more properties of a complex object so as to attend to others

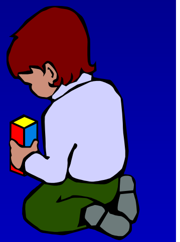
=> Remove detail (simplify) and focus (selection)

- a general concept formed by extracting common features from specific examples
- the process of formulating general concepts by abstracting common properties of instances

=> generalisation (core or essence)

Models and Modelling?

- A model is a description from which detail has been removed in a systematic manner and for a particular purpose.
- A simplification of reality intended to promote understanding.
- Models are the most important engineering tool; they allow us to understand and analyse large and complex problems.



Questions

- For the task at hand, how is the "right" level of abstraction selected?
  What heuristics can be used to decide what concepts should be left out of a modeling language?
- How can we measure, test, and teach abstraction skills suitable for modelling?

More Questions

- To what extent to do domain-specific modeling language approaches provide mechanisms for extending modeling languages with support for new abstractions?
  How do domain-specific modeling languages offer advantages over UML; likewise, what advantages remain in using UML over customized modeling languages?
- In terms of providing the best constructs for abstraction in modeling languages, what can be learned from decades of programming language design (if anything)?

And yet another Question

- What are examples of cases where the LACK of abstraction in modeling hindered a project?

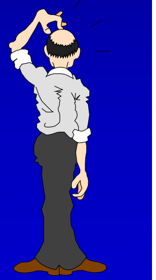
  What was missing in the modeling language and how can the language be extended to address new constructions for the abstractions needed for these examples?

Questions

- For the task at hand, how is the "right" level of abstraction selected?

  What heuristics can be used to decide what concepts should be left out of a modeling language?

- Properties of interest – "fit for purpose"
- support for analysis and reasoning
- permit you to frame questions of interest
- appropriate for the particular phase in software development
- Not easy…!



Ockam's Razor

- William of Ockam (1285) formulated the famous "Rule of the Razor":

  Entia non sunt multiplicanda sine necessitate.

  Entities should not be multiplied without necessity.

- In other words a model should be as simple as possible, but no simpler - it should discard elements of no interest.
- "Fit for purpose".

Questions

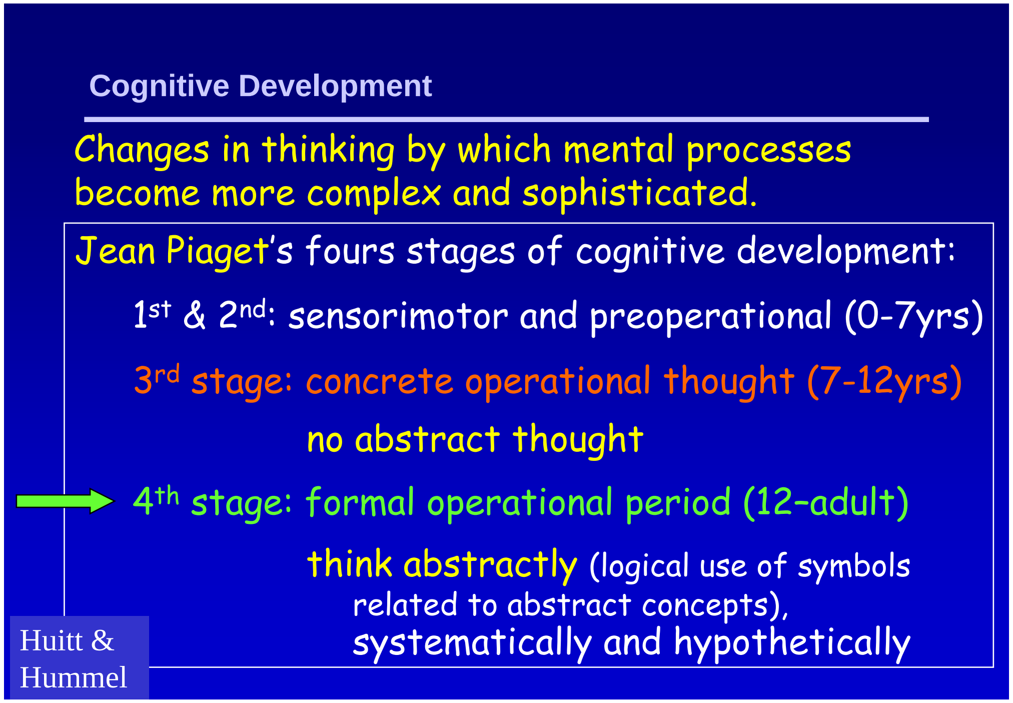
- How can we measure, test, and teach abstraction skills suitable for modelling?

- Cognitive development provides some guidelines
- Tests for Formal Operational Stage
- Have been studies.
- None exist which focus specifically on abstraction skills

- Potential benefit by improving student selection, checking progress, checking education effectiveness, …

If you are interested in more …..

- Is abstraction the key to computing?
  Communications of the ACM
  Volume 50 , Issue 4 (April 2007), Pages: 36-42



Cognitive Development

Changes in thinking by which mental processes become more complex and sophisticated.

Jean Piaget's fours stages of cognitive development:

1st & 2nd: sensorimotor and preoperational (0-7yrs)

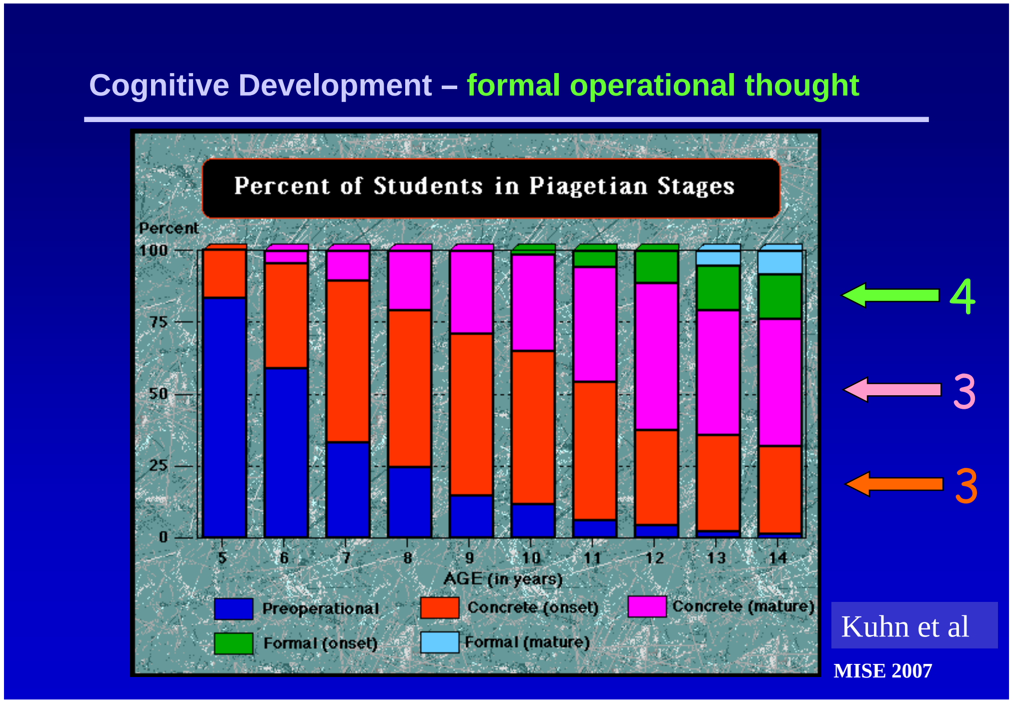
3rd stage: concrete operational thought (7-12yrs)

no abstract thought

→ 4th stage: formal operational period (12-adult)

think abstractly (logical use of symbols related to abstract concepts), systematically and hypothetically

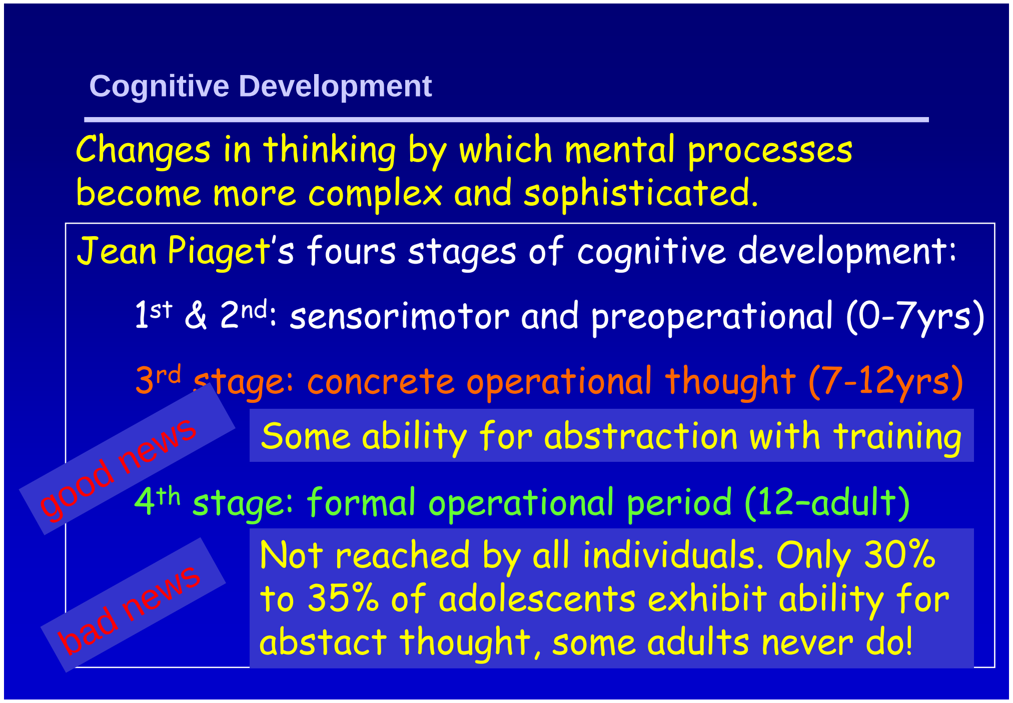
Huitt & Hummel



Cognitive Development – formal operational thought

Kuhn et al

MISE 2007



Cognitive Development

Changes in thinking by which mental processes become more complex and sophisticated.

Jean Piaget's fours stages of cognitive development:

1st & 2nd: sensorimotor and preoperational (0-7yrs)

3rd stage: concrete operational thought (7-12yrs)

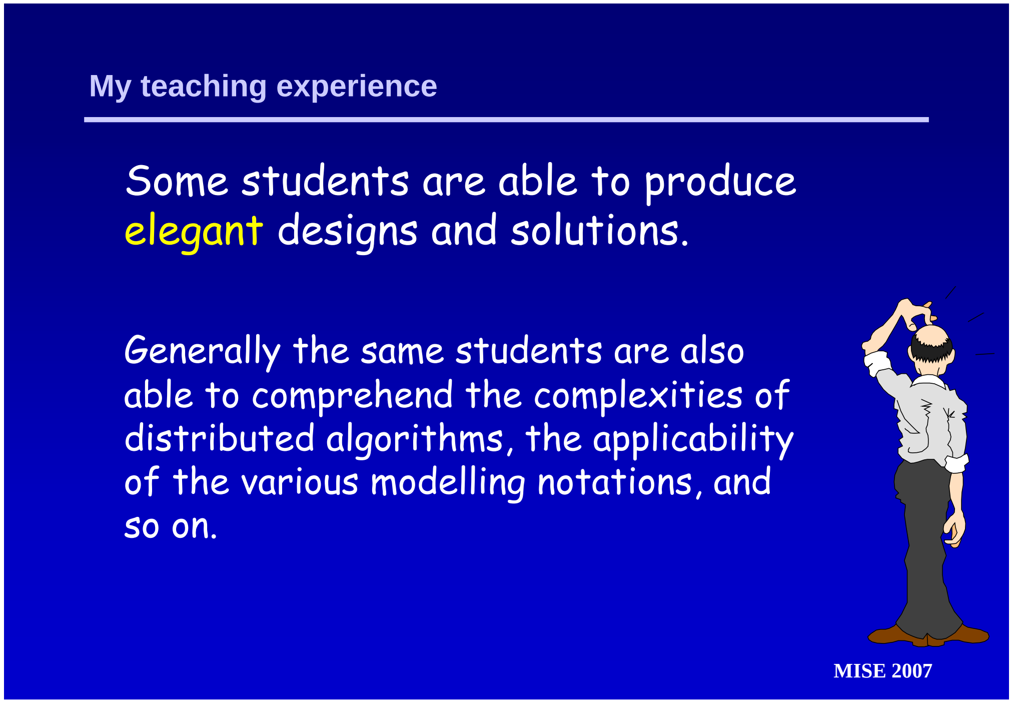
Some ability for abstraction with training

good news

4th stage: formal operational period (12-adult)

Not reached by all individuals. Only 30% to 35% of adolescents exhibit ability for abstact thought, some adults never do!

bad news



My teaching experience

Some students are able to produce elegant designs and solutions.

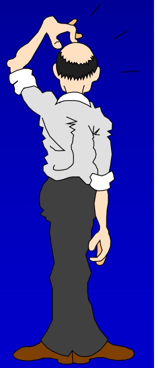
Generally the same students are also able to comprehend the complexities of distributed algorithms, the applicability of the various modelling notations, and so on.

MISE 2007

Experience: the others ....

A number of others are not so able.

They tend to find distributed algorithms very difficult, do not appreciate the utility of modelling, find it difficult to know what is important in a problem, produce convoluted solutions which replicate the problem complexities, ……



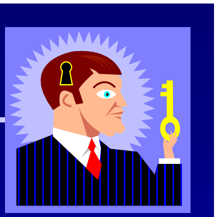
Why ?

I believe .....

… that the heart of the problem lies in a difficulty in dealing with

Abstraction

Abstraction – the key to Software Engineering?



If we want the best Software Engineers, we need to …

- teach them abstraction skills
- perhaps we should consider selecting students for Computing based not only on their school grades, but also on their abstraction abilities?

i.e. Perhaps we should test their ability for formal operational thinking?