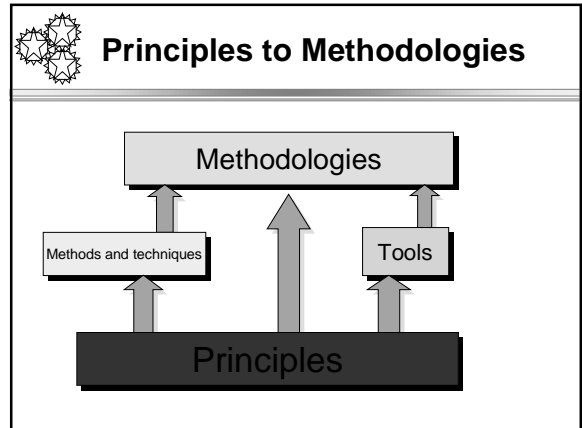


Software Engineering Principles



- ## Seven Principles
- Rigor and Formality
 - Separation of Concerns
 - Modularity
 - Abstraction
 - Anticipation of Change
 - Generality
 - Incrementality

- ## Rigor and Formality
- Rigor is the “tightness” of the definition, design, statement, etc.
 - Formality always based on mathematical laws
- Sloppy \implies Rigorous \implies Formal

Rigor vs. Formality

“make z bigger than x or y”	<i>Sloppy</i>
“Pick the bigger one”	<i>fairly rigorous</i>
“Set z to the larger of x or y”	<i>more rigorous</i>
	<i>formal</i>
$z = \max(x,y)$	<i>formal, and simpler</i>

Warning: Formality can obscure the problem where appropriate rigor can help

- ## Separation of Concerns
- Pull different parts of the problem apart
 - Appears simple, but deceptively hard
 - *Divide and Conquer* is a SoC strategy
 - Partitioning strategies can make or break
 - time => schedules
 - concepts => user interface vs. algorithmic code
 - process => generating tasks

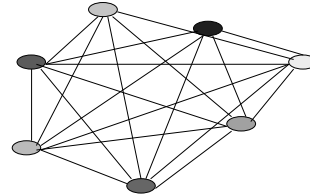


Modularity

- Divide complexity into simpler, but rational, pieces
- Related to SoC
- Complexity reduction
 - Increases chances of understanding complexity
- Most modern technology built on this principles
- Raises the issue of interface



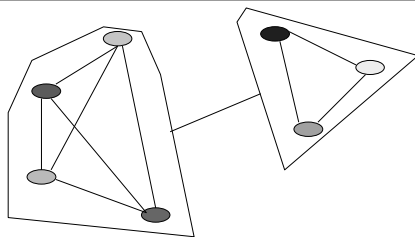
Complexity Reduction



Got all interactions? $6+5+4+3+2+1=21$



Effect of Simple Module



$$6+1+3=10$$



Abstraction

- Stripping away what's not important
- Implies *KEEPING* what is important
- Essential principle in modeling
- Takes practice -- Few people do it well.



Anticipation of Change

- This applies to the software product, not the methods (usually... see? *I'm anticipating change*)
- It's going to change, so build it in from the start
 - Not free... takes effort and cost
 - ◆ This implies there is a trade-off between flexibility and cost/effort
- Minimum: Wandering requirements
- Maximum: Reusable components



Generality

- "Backup" to a "higher" view to find the bigger problem that covers this problem
- Plays into modularity
 - Modularity uses generalization to group functions
- Draws on abstraction
 - Have to see the essentials of the problem
- As usual, requires trade-offs



Incrementality

- The idea of proceeding in steps
- Requires separation of concerns to break problem down into steps
- Evolutionary approach to design
 - prototypes